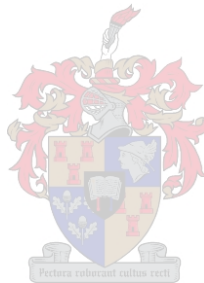


Optimal Management of MPLS Networks

Johannes Marthinus de Kock



Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Science
at the University of Stellenbosch

Supervisor: Prof. A. E. Krzesinski

March 2002

Declaration

I the undersigned hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Signature:

Date:

Abstract

Multiprotocol Label Switching (MPLS) is a routing technology which can manage Quality of Service (QoS) in scalable connectionless networks using relatively simple packet forwarding mechanisms. This thesis considers the optimisation of the QoS offered by an MPLS network. The QoS measure used is the expected packet delay which is minimised by switching packets along optimal label switched paths (LSPs).

Two mathematical models of MPLS networks are presented together with appropriate algorithms for optimally dividing the network traffic into forwarding equivalence classes (FECs), finding optimal LSPs which minimise the expected packet delay and switching these FECs along the optimal LSPs. These algorithms are applied to compute optimal LSPs for several test networks. The mathematics on which these algorithms are based is also reviewed.

This thesis provides the MPLS network operator with efficient packet routing algorithms for optimising the network's QoS.

Opsomming

Multiprotocol Label Switching (MPLS) is 'n roeteringsmetode om die diensvlak (QoS) van 'n skaleerbare, verbindinglose netwerk te bestuur deur middel van relatief eenvoudige versendingsmeganismes. Hierdie tesis beskou die optimering van die QoS van 'n MPLS-netwerk. Die QoS-maatstaf is die verwagte vertraging van 'n netwerk-pakkie. Dit word geminimeer deur pakkies langs optimale "label switched paths" (LSPs) te stuur.

Twee wiskundige modelle van MPLS-netwerke word ondersoek. Toepaslike algoritmes word verskaf vir die optimale verdeling van die netwerkverkeer in "forwarding equivalence classes" (FECs), die soektog na optimale LSPs (wat die verwagte pakkie-vertraging minimeer) en die stuur van dié FECs langs die optimale LSPs. Hierdie algoritmes word ingespan om optimale LSPs vir verskeie toetsnetwerke op te stel. Die wiskundige teorie waarop hierdie algoritmes gegrond is, word ook hersien.

Hierdie tesis verskaf doeltreffende roeteringsalgoritmes waarmee 'n MPLS-netwerkbestuurder/-es die netwerk se QoS kan optimeer.

Acknowledgements

This work was performed within the *Siemens-Telkom Centre of Excellence for ATM & Broadband Networks and their Applications* and is supported by grants from the South African National Research Foundation, Telkom SA Limited and Siemens Telecommunications.

I also spent a sabbatical at the *Teletraffic Research Centre* at the University of Adelaide. My host was Prof PG Taylor from the Department of Applied Mathematics at the University of Adelaide.

List of Publications

1. SA Berezner, AM Ingg, JM de Kock and AE Krzesinski. Alternative Routing and Reconfiguration in Communication Networks. In Proceedings of *TeleTraffic '97*, Grahamstown, South Africa, September 1997.
2. JM de Kock and AE Krzesinski. Computing an Optimal Virtual Path Connection Network by Simulated Annealing. In Proceedings of the *South African Telecommunications, Networks and Applications Conference (SATNAC) '98*, Cape Town, South Africa, September 1998.
3. JM de Kock and AE Krzesinski. Dealing with Instability in the Reduced Load Approximation. In Proceedings of the *South African Telecommunications, Networks and Applications Conference (SATNAC) '99*, Durban, South Africa, September 1999.
4. Å Arvidsson, JM de Kock, AE Krzesinski and PG Taylor. Cost-Effective Deployment of Bandwidth Partitioning in Broadband Networks. In Proceedings of the *South African Telecommunications, Networks and Applications Conference (SATNAC) '99*, Durban, South Africa, September 1999.
5. SA Berezner, JM de Kock, AE Krzesinski and PG Taylor. Local Reconfiguration of ATM Virtual Path Connection Networks. In Proceedings of *IFIP Broadband Communications '99*, Hong Kong, November 1999.
6. SA Berezner, JM de Kock and AE Krzesinski. The Design of Optimal Virtual Path Connection Networks with Service Separation. In Proceedings of the *International Conference on Information, Communications and Signal Processing (ICICS) '99*, Singapore, December 1999.
7. JM de Kock and AE Krzesinski. Quality of Service Optimisation in MPLS Networks. In Proceedings of the *IFIP Workshop on Performance Modelling and Evaluation of ATM & IP Networks (IFIP ATM&IP) 2000*, Ilkley, UK, July 2000.
8. JM de Kock and AE Krzesinski. Optimising the Quality of Service in MPLS Networks. In Proceedings of the *South African Telecommunications, Networks and Applications Conference (SATNAC) 2000*, Somerset West, South Africa, September 2000.

9. Å Arvidsson, JM de Kock, AE Krzesinski and PG Taylor. The Design of ATM Virtual Path Connection Networks with Service Separation. In Proceedings of the *South African Telecommunications, Networks and Applications Conference (SATNAC) 2000*, Somerset West, South Africa, September 2000.
10. JM de Kock and AE Krzesinski. Finding Optimal Paths in MPLS Networks. In Proceedings of *Africom 2001*, Cape Town, South Africa, May 2001.
11. JE Burns, TJ Ott, JM de Kock and AE Krzesinski. Path Selection and Bandwidth Allocation in MPLS Networks: a Non-linear Programming Approach. In Proceedings of *ITCom 2001*, Denver, Colorado, USA, August 2001.

Contents

Abstract	v
Opsomming	vii
Acknowledgements	ix
List of Publications	xi
1 Introduction	1
1.1 Label Switching	1
1.2 Multiprotocol Label Switching	2
1.3 Flow Deviation	3
2 The Flow Deviation Algorithm	5
2.1 The Model	5
2.2 The Optimisation Problem	7
2.3 The Kleinrock Algorithm	12
2.4 The Bertsekas-Gallager Algorithm	16
2.5 Applying Flow Deviation to MPLS Networks	20
3 Minimising Nodal Delays in MPLS Networks	21
3.1 The Model	21
3.2 The Expected Packet Delay in MPLS Networks	23

3.3	The LSR's Queueing Discipline	25
3.4	Finding Optimal LSPs	26
3.4.1	Initial Feasible Flow Vector	30
3.5	Some Applications	33
3.5.1	Two Small Networks	34
3.5.2	A Larger Network	35
3.6	Conclusion	36
4	Minimising Link Delays in MPLS Networks	37
4.1	The Model	37
4.2	Analysis of the Link Delays	38
4.3	A Quantitative Measure of the Link Delay	40
4.4	Finding Optimal Label Switched Paths	40
4.5	Results	45
4.5.1	The Convergence of the Flow Deviation Algorithms	45
4.5.2	The LSP Sets \mathcal{R}	46
4.6	Conclusion	60
5	Conclusion	61
A	Convex and Concave Functions	63
B	The Frank-Wolfe Method	67
B.1	Some Results from Linear Algebra	67
B.2	Some Results from Optimisation Theory	70
B.3	The Frank-Wolfe Method	79
C	The Fibonacci Search Method	87
D	Newton's Method	91

CONTENTS	xv
<hr/>	
D.1 The Newton-Raphson Method	91
D.2 A Cauchy-Type Steepest Descend Method	93
D.3 A Gradient Projection Method	94
Bibliography	97

List of Tables

3.1	The arrival and service rates for the 4 node network	34
3.2	The packet arrival rates for the SA network	35
3.3	The LSPs used by the SA network	36
4.1	LSP Correlation	47
4.2	LSP Statistics of the Kleinrock Algorithm for the 50 Node Network	49
4.3	LSP Statistics of the Bertsekas-Gallager Algorithm for the 50 Node Network . . .	51
4.4	Normalised LSP Statistics of the Kleinrock Algorithm for the 50 Node Network . .	51
4.5	Normalised LSP Statistics of the Bertsekas-Gallager Algorithm for the 50 Node Network	52
4.6	LSP Multiplicity of the Kleinrock Algorithm for the 50 Node Network	52
4.7	LSP Multiplicity of the Bertsekas-Gallager Algorithm for the 50 Node Network . .	52
4.8	LSP Statistics of the Kleinrock Algorithm for the 100 Node Network	55
4.9	LSP Statistics of the Bertsekas-Gallager Algorithm for the 100 Node Network . . .	58
4.10	Normalised LSP Statistics of the Kleinrock Algorithm for the 100 Node Network .	59
4.11	Normalised LSP Statistics of the Bertsekas-Gallager Algorithm for the 100 Node Network	59
4.12	LSP Multiplicity of the Kleinrock Algorithm for the 100 Node Network	59
4.13	LSP Multiplicity of the Bertsekas-Gallager Algorithm for the 100 Node Network .	60

List of Figures

2.1	The Route Sets	6
3.1	The Set \mathcal{R}_j	22
3.2	The Set \mathcal{B}_n	23
3.3	A 6 node network	34
3.4	A 4 node network	34
3.5	The SA MPLS network	35
4.1	The Topology of the 10 Node Network	45
4.2	The Topology of the 20 Node Network	45
4.3	Convergence for the 10 Node Network	46
4.4	Convergence of the Bertsekas-Gallager Algorithm for the 10 Node Network	47
4.5	Convergence for the 20 Node Network	48
4.6	Convergence of the Bertsekas-Gallager Algorithm for the 20 Node Network	49
4.7	LSP Correlation for the 10 Node Network	50
4.8	LSP Correlation for the 20 Node Network	53
4.9	The Topology of the 50 Node Network	54
4.10	The Topology of the 100 Node Network	55
4.11	LSP Correlation for the 50 Node Network	56
4.12	LSP Correlation for the 100 Node Network	57
4.13	Percentage use of Shortest Possible LSPs	58
A.1	A Convex Function	63

A.2	A Concave Function	64
B.1	Convex Feasible Region \mathcal{G} Formed by Linear Constraints	75
B.2	A Supporting Line P_x	76
B.3	A Supporting Hyperplane $P_{\mathbf{x}}$	77
B.4	A Non-Optimal Point \mathbf{x}	78
B.5	An Optimal Point \mathbf{x}_0	79

Chapter 1

Introduction

The Quality of Service (QoS) offered by a *connection-oriented* network can be managed by optimally utilising the transmission capacity or bandwidth of the underlying physical network. There are several possible criteria of optimality, for example network throughput, blocking probability or rate of earning revenue. Several approaches to bandwidth management (and thus QoS management) have been discussed in the literature. These include VP (virtual path) distribution algorithms [2], VPCN (virtual path connection network) optimisation methods [5, 7, 8] and the design of virtual subnetworks [15].

The success of the Internet is making *connectionless* networks based on the Internet Protocol (IP) increasingly popular. Current generation IP networks do not provide effective mechanisms (apart from priority queueing flags in packets) for managing QoS. This problem was eventually addressed by several vendors including Toshiba, IBM, Ipsilon and Cisco. Each vendor came up with some version of a technology now known as *label switching*. By allowing explicit routing, label switching enables network operators to manage QoS by means of optimal routing.

1.1 Label Switching

A label is a field in an IP packet that is used to determine the route followed by a packet. A label switching network consists of a group of interconnected *label switching routers* (LSRs). An LSR performs *label swapping* on incoming packets. During label swapping an incoming packet's local (or global) label is examined and replaced by an appropriate global (or local) label. Whether a local label is replaced by a global label or vice versa depends on the type of *label binding* ("downstream" or "upstream") used. A packet is typically created by an application on a computer in a subnetwork connected to an LSR referred to as the *ingress LSR*. The ingress LSR is also referred to as the originating node (ON) (see Ash *et al.* [1], for example). Upon receiving the unlabeled packet, the ingress LSR assigns a label to the packet and forwards the packet to the next LSR on the

packet's route. The LSR uses the information in the label and in the LSR routing table, or *label information base* (LIB) to identify the next LSR on the packet's route. The other LSRs traversed by the packet's route perform label swapping on the incoming packet and forward the packet to the next node on its route. Finally, the last LSR on the packet's route, known as the *egress LSR*, removes the label and passes the packet to the appropriate application in the subnetwork connected to it.

Label switching networks can use either *destination-based* or *explicit* routing. The routing decision in destination-based is based only on the packet's destination address. In the case of explicit routing the route is specified by the packet's label. The route which a packet follows through the label switching network is known as a *label switched path* (LSP). We do not allocate bandwidth to an LSP. Thus we neither set up *trunks*, nor do we construct *constraint-based routing label switched paths* (CRLSP) as mentioned in [1].

A label switching network using explicit routing allows *fine forwarding granularity* – the set of packets which an LSR can receive is partitioned into disjoint subsets known as *forwarding equivalence classes* (FECs). The set of packets belonging to a particular service class and travelling between a given O-D pair can be assigned to a distinct FEC or the FEC that a packet belongs to can be based on the computer in the subnetwork where the packet originated and/or (corresponding to even finer forwarding granularity) the application on the computer which generated the packet. Fine forwarding granularity makes the network more flexible since it allows routing based on service class [13]. Ash *et al.* [1] divides all path selection methods into four categories, namely *hierarchical fixed routing* (FR), *timedependent routing* (TDR), *state-dependent routing* (SDR) and *event-dependent routing* (EDR). Our model of explicit routing is such that FR, TDR and SDR are all modelled. EDR is the only path selection category not covered since EDR routing tables are updated locally, whereas ours is a centralised approach.

This thesis examines how fine forwarding granularity can be used to optimise the network's QoS. The *expected packet delay* is used as the network's performance criterion. Fine forwarding granularity enables the network operator to minimise the expected packet delay. This is done by allowing several LSPs between the same O-D pair. The total packet load offered to an O-D pair is switched among several LSPs in such a way that the expected packet delay is minimised. Thus the set of all packets offered to an O-D pair is partitioned into FECs and the packets in each FEC are switched along a particular LSP.

1.2 Multiprotocol Label Switching

The label switching protocol used in this thesis is *Multiprotocol Label Switching* (MPLS) which was introduced by the Internet Engineering Task Force (IETF). As the name indicates, it combines the label switching approaches of the four vendors mentioned above. The expected packet delay is minimised in MPLS networks with explicit routing.

The expected packet delay depends on the delays in the links (due to bandwidth limitations) and the delays in the nodes LSRs (due to the limited speed with which an LSR can copy a packet on an outgoing link). This thesis considers these scenarios separately.

In the first scenario the link bandwidths are assumed to be infinite. It has been predicted that the decreasing cost of optical fibre and other transmission media will provide future networks with nearly unlimited bandwidth. If this “infinite bandwidth” model is applied to an IP network, the network’s QoS is no longer influenced by the availability of sufficient transmission capacity. The *nodal delays* due to the service and queueing of the packets in the LSRs will primarily determine the QoS offered by the network. We therefore consider the expected packet delay to be composed solely of the nodal delays. Chapter 3 covers this scenario.

In the second (more realistic scenario) we consider the links to have finite bandwidth and the major source of delay consists of the queueing and serving of the packets at the links. The *link delays* dominate to such an extent that the nodal delays are considered negligible. Chapter 4 considers this scenario.

1.3 Flow Deviation

The object of the label switching approach is different from the ATM network case. In an ATM network the performance criterion is the expected *rate of earning revenue* (or profit) whereas the expected *packet delay* is the performance criterion in an MPLS network. An ATM network operator can achieve this by constructing a fully-meshed VPCN which maximises the network’s expected rate of earning revenue. On the other hand, the operator of an MPLS network switches packet flows onto optimal LSPs in order to minimise the expected packet delay. In the label switching scenario considered in this thesis *flow deviation* is performed rather than *capacity reservation* – no CRLSP *virtual network* (VNET) (the MPLS equivalent of a VPCN, see [1]) is constructed but IP packets are switched along appropriate LSPs. Chapter 2 considers the origin and different versions of the flow deviation algorithm. The mathematical methods on which the flow deviation algorithms are based are reviewed in the appendices.

Chapter 2

The Flow Deviation Algorithm

This chapter discusses the *flow deviation* algorithm in abstract terms. The basis of the discussion is a general *label switching network* with a *cost function*. The precise meaning of a general label switching network is given in section 2.1. Only the properties of the cost function are specified and neither the function nor its physical interpretation (eg delay) is fixed. Section 2.1 introduces the network model and section 2.2 states the optimisation problem to be solved by flow deviation. Sections 2.3 and 2.4 discuss two variants of the flow deviation algorithm.

2.1 The Model

The network has N nodes where each node receives packets on incoming links and forwards them on appropriate outgoing links. The nodes are numbered from one and identified by their numbers. Let $\mathcal{N} = \{1, 2, \dots, N\}$ denote the set of nodes. Each O-D pair is assigned a unique integer. Let \mathcal{J} be the set of all O-D pairs (identified by their numbers). Therefore $J = |\mathcal{J}| = N(N - 1)$. Each physical link corresponds to an O-D pair, consequently the links are identified by the numbers of the O-D pairs which they connect. Let \mathcal{L} denote the set of links (identified by their link numbers) and let $L = |\mathcal{L}|$ then $\mathcal{L} \subseteq \mathcal{J}$ and $L \leq J$. The function $\mathcal{L} : \mathcal{N} \times \mathcal{N} \mapsto \mathcal{J}$ is defined as $\mathcal{L}(o, d) = j$ if O-D pair (o, d) is numbered j .

Let C_ℓ denote the (physical) capacity (zero if the physical link does not exist) of the uni-directional link $o - d$ for each O-D pair (o, d) such that $\mathcal{L}(o, d) = \ell$. Thus the set \mathcal{L} can be expressed as

$$\mathcal{L} = \{j \in \mathcal{J} \mid C_j > 0\}.$$

Let λ_j denote the arrival rate of packets to O-D pair j and let λ be the total packet arrival rate to the network:

$$\lambda = \sum_{j=1}^J \lambda_j.$$

λ is also referred to as the total *external traffic* attempting to enter the network.

A *route* $r = \langle o, d \rangle$ is a sequence of physical links $o - o_1, o_1 - o_2, \dots, o_m - d$ connecting nodes o and d . $r = \langle o, d \rangle$ can also consist of the single link $o - d$. Note the notation distinction: (o, d) denotes the O-D pair, $o - d$ denotes the physical link and $\langle o, d \rangle$ denotes a route connecting nodes o and d . Let \mathcal{R}_j denote the set of routes between O-D pair j . Figure 2.1 shows the set $\mathcal{R}_j = \{r_1, r_2\}$ where $\mathcal{L}(o_1, d_1) = j$. Let \mathcal{R} be the set of all routes in the network:

$$\mathcal{R} = \bigcup_{j \in \mathcal{J}} \mathcal{R}_j.$$

Let \mathcal{A}_ℓ denote the set of routes that traverse link ℓ for each $\ell \in \mathcal{L}$. The set $\mathcal{A}_\ell = \{r_3, r_4, r_5\}$ is shown in figure 2.1. The routes r_3, r_4 and r_5 all traverse link $o_2 - d_2$ with $\mathcal{L}(o_2, d_2) = \ell$.

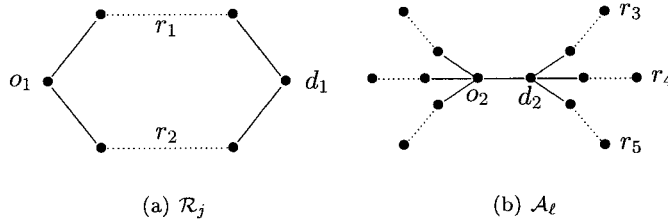


Figure 2.1: The Route Sets

All packets offered to O-D pair j are switched along routes in \mathcal{R}_j . Let s_{rj} be the portion of the packets offered to O-D pair j which travel along route r . If ν_r is the arrival rate of packets to route r , then $\nu_r = s_{rj} \lambda_j$ for all $j \in \mathcal{J}$ and $r \in \mathcal{R}_j$. All the packets offered to O-D pair j are switched along routes in \mathcal{R}_j , therefore the following two related equations hold

$$\lambda_j = \sum_{r \in \mathcal{R}_j} \nu_r \tag{1}$$

$$\sum_{r \in \mathcal{R}_j} s_{rj} = 1 \tag{2}$$

for all $j \in \mathcal{J}$.

The network model specified here is that of a general label switching network. The standard label switching notions such as packets, switching, LSPs and FECs are incorporated in this model. LSPs are referred to as routes to keep the discussion general. FECs are not explicitly mentioned but are implied by the portions $s_{r,j}$. The main object of this thesis is the optimal management of MPLS networks. The management goals can be specified in terms of a non-linear optimisation problem. The next section discusses an optimisation problem which specifies the optimal management of the general label switching network which we are considering in this chapter.

2.2 The Optimisation Problem

This section introduces the general optimisation problem which is adapted to describe specific MPLS network models in chapters 3 and 4.

Let γ_ℓ denote the rate at which packets arrive to link ℓ . The relationship between γ_ℓ and ν_r is

$$\gamma_\ell = \sum_{r \in \mathcal{A}_\ell} \nu_r \quad \text{for all } \ell \in \mathcal{L}. \quad (3)$$

A packet arrival rate will be referred to as a *flow* in the remainder of this chapter. Accordingly, (1) and (2) are known as *preservation of flow* equations since they specify that the total flow in the network is preserved.

Let D_ℓ be the *cost function* for link ℓ . Some general properties of D_ℓ are specified but not D_ℓ itself. D_ℓ denotes a class of functions of which the individual members describe particular label switching networks. The most important property is that D_ℓ is a function of the rate γ_ℓ (the *link flow*) at which packets are offered to link ℓ . D_ℓ is also a function of a fixed service rate in chapter 3 and a function of the capacity C_ℓ in chapter 4. However, the capacities and service rates are regarded as constants for a given network. Thus we consider D_ℓ as a function of a single variable γ_ℓ and the derivatives of D_ℓ with respect to γ_ℓ are considered as ordinary derivatives as opposed to partial derivatives. We make an important assumption about D_ℓ .

Assumption 2.1 (Differentiability of D_ℓ). D_ℓ is a differentiable function of γ_ℓ and is defined on the interval $[0, C_\ell]$ for each $\ell \in \mathcal{L}$.

Let D be the cost function for the network,

$$D(\gamma) = \sum_{\ell=1}^L D_\ell(\gamma_\ell), \quad (4)$$

where $\gamma = (\gamma_\ell)_{\ell \in \mathcal{L}}$ is the *link flow vector*. Define $\nu = (\nu_r)_{r \in \mathcal{R}}$ as the *route flow vector*. The cost function can be written as (using (3))

$$D(\nu) = \sum_{\ell=1}^L D_{\ell} \left(\sum_{r \in \mathcal{A}_{\ell}} \nu_r \right). \quad (5)$$

The object of this chapter is to solve the following non-linear optimisation problem.

Optimisation Problem 2.1.

Minimise:

$$D(\nu) = \sum_{\ell=1}^L D_{\ell} \left(\sum_{r \in \mathcal{A}_{\ell}} \nu_r \right)$$

Subject to:

$$\begin{aligned} \lambda_j &= \sum_{r \in \mathcal{R}_j} \nu_r && \text{for all } j \in \mathcal{J} \\ \nu_r &\geq 0 && \text{for all } r \in \mathcal{R}. \end{aligned}$$

We introduce an important assumption before discussing the solution.

Assumption 2.2 (Flow Independence). *The γ_{ℓ} 's associated with different links are independent and the ν_r 's associated with different routes are independent.*

Define a *cost rate vector* $\mathbf{c} = (c_1, c_2, \dots, c_L)$ where

$$c_{\ell} = \frac{\partial D}{\partial \gamma_{\ell}}.$$

c_{ℓ} is the rate at which the network cost D increases with an infinitesimal increase in the link flow γ_{ℓ} . Thus, if the flow along some route r is increased such that the flow on link ℓ increases by an infinitesimal amount σ_{ℓ} , the increase in D is $\delta D = \sigma_{\ell} c_{\ell}$. The vector \mathbf{c} is the gradient vector of D at the point γ which is denoted by $\nabla D(\gamma)$. Note that an important consequence of the flow independence assumption is that

$$c_{\ell} = \frac{\partial D}{\partial \gamma_{\ell}} = \frac{dD_{\ell}}{d\gamma_{\ell}} = D'_{\ell}.$$

Let $r \in \mathcal{R}_j$ and define the cost of route r as

$$c_r = \frac{\partial D}{\partial \nu_r}$$

then

$$\begin{aligned}
 c_r &= \frac{\partial}{\partial \nu_r} \sum_{\ell=1}^L D_\ell \\
 &= \sum_{\ell=1}^L \frac{dD_\ell}{d\gamma_\ell} \frac{\partial \gamma_\ell}{\partial \nu_r} \\
 &= \sum_{\ell \in r} D'_\ell \\
 &= \sum_{\ell \in r} c_\ell
 \end{aligned} \tag{6}$$

which is obtained by differentiating (4), applying the flow independence assumption and noting that (3) implies

$$\frac{\partial \gamma_\ell}{\partial \nu_r} = \begin{cases} 1 & \text{if } \ell \in r \\ 0 & \text{otherwise.} \end{cases}$$

The *route cost* c_r is also known as the *first derivative length* of a route. Consider a particular O-D pair j . A route $r_j \in \mathcal{R}_j$ for which the route cost is minimal (ie $c_{r_j} = \min \{c_r \mid r \in \mathcal{R}_j\}$) is known as a *least-cost route* connecting O-D pair j . A route $r \in \mathcal{R}_j$ for which c_r is not minimal is referred to as an *extremal route*. Similarly, the flows on the least-cost routes and the extremal routes are referred to as *least-cost flows* and *extremal flows* respectively.

Let $\tilde{\nu} = (\tilde{\nu}_r)_{r \in \mathcal{R}}$ be an optimal route flow vector (ie an optimal solution to Optimisation Problem 2.1). If $\tilde{\nu}_{r_j} > 0$ for some $r_j \in \mathcal{R}_j$, an infinitesimal amount of flow $\delta \nu_{r_j} > 0$ can be shifted from r_j to some route $r \in \mathcal{R}_j$ without decreasing the cost D . The change δD in cost is

$$\delta D = \delta \nu_{r_j} \frac{\partial}{\partial \nu_r} D(\tilde{\nu}) - \delta \nu_{r_j} \frac{\partial}{\partial \nu_{r_j}} D(\tilde{\nu})$$

and since δD must be non-negative

$$\begin{aligned}
 \delta \nu_{r_j} \frac{\partial}{\partial \nu_r} D(\tilde{\nu}) &\geq \delta \nu_{r_j} \frac{\partial}{\partial \nu_{r_j}} D(\tilde{\nu}) \\
 \frac{\partial}{\partial \nu_r} D(\tilde{\nu}) &\geq \frac{\partial}{\partial \nu_{r_j}} D(\tilde{\nu})
 \end{aligned}$$

This implies that if $\tilde{\nu}_{r_j} > 0$ for $r_j \in \mathcal{R}_j$, then

$$\frac{\partial}{\partial \nu_r} D(\tilde{\nu}) \geq \frac{\partial}{\partial \nu_{r_j}} D(\tilde{\nu}) \quad \text{for all } r \in \mathcal{R}_j \quad (7)$$

and

$$\frac{\partial}{\partial \nu_{r_j}} D(\tilde{\nu}) = \min \left\{ \frac{\partial}{\partial \nu_r} D(\tilde{\nu}) \mid r \in \mathcal{R}_j \right\}$$

or alternatively $c_{r_j} = \min \{c_r \mid r \in \mathcal{R}_j\}$ which identifies r_j as a least-cost route. Thus an optimal route flow $\tilde{\nu}_{r_j}$ is positive only on a route r_j with a minimal first derivative length. Furthermore, at an optimal route flow vector, the routes among which λ_j is split must have equal first derivative lengths (ie they must all be least-cost routes). Thus (7) is a necessary condition for optimality. If the cost function D is convex, (7) is also a sufficient condition for optimality (see Bertsekas *et al.* [9]).

We define a route flow vector as *feasible* if the constraints of Optimisation Problem 2.1 are satisfied. Thus we have the following definition.

Definition 2.1 (Feasible Route Flow Vector). *A route flow vector ν is feasible if*

$$\lambda_j = \sum_{r \in \mathcal{R}_j} \nu_r \quad \text{for all } j \in \mathcal{J} \quad (8)$$

$$\nu_r \geq 0 \quad \text{for all } r \in \mathcal{R}. \quad (9)$$

Given a feasible route flow vector ν , consider changing ν along a direction $\Delta\nu = (\Delta\nu_r)_{r \in \mathcal{R}}$ and thus obtaining the route flow vector $\nu + \alpha\Delta\nu$.

Bertsekas *et al.* [9] mention three requirements which $\Delta\nu$ has to meet.

1. The first requirement is the feasibility of the resulting route flow vector. Formally this means that for some $\alpha_{\max} > 0$ and any $\alpha \in [0, \alpha_{\max}]$, the flow vector $\nu + \alpha\Delta\nu$ must be feasible. Thus by (8)

$$\lambda_j = \sum_{r \in \mathcal{R}_j} (\nu_r + \alpha\Delta\nu_r)$$

and since ν is also feasible,

$$0 = \sum_{r \in \mathcal{R}_j} \alpha\Delta\nu_r \quad \text{and therefore} \quad 0 = \sum_{r \in \mathcal{R}_j} \Delta\nu_r$$

for all $j \in \mathcal{J}$. Similar to (1) and (2), this also implies the preservation of flow.

2. The second requirement for feasibility (9) implies that $\nu_r + \alpha \Delta \nu_r \geq 0$ for all $r \in \mathcal{R}$ and thus $\Delta \nu_r \geq 0$ for all $r \in \mathcal{R}$ such that $\nu_r = 0$.
3. The final requirement is that $\Delta \nu$ is a descent direction and therefore the cost function D can be decreased by making small moves from ν in the direction $\Delta \nu$. Let $\nabla D(\nu)$ denote the gradient vector of D at the point ν . The requirement that $\Delta \nu$ should be a descent direction implies that

$$\nabla D(\nu) \cdot \Delta \nu < 0.$$

The requirements on the the direction $\Delta \nu$ are satisfied by a family of iterative algorithms that do the following. Let the current route flow vector (at iteration m) be $\nu^m = (\nu_r^m)_{r \in \mathcal{R}}$. The route flow vector ν^m is now changed to

$$\nu_r^{m+1} = (1 - \alpha_r) \nu_r^m + \alpha_r \delta \nu_r \quad (10)$$

where $\alpha_r \in [0, 1]$ for each $r \in \mathcal{R}$ and $\delta \nu$ is a feasible route flow vector. The direction $\Delta \nu$ at iteration m is now given by

$$\Delta \nu = \delta \nu - \nu^m.$$

Each of these algorithms ensures that the preservation of flow condition is met

$$\begin{aligned} \sum_{r \in \mathcal{R}_j} \Delta \nu_r &= \sum_{r \in \mathcal{R}_j} (\delta \nu_r - \nu_r^m) \\ &= \sum_{r \in \mathcal{R}_j} \delta \nu_r - \sum_{r \in \mathcal{R}_j} \nu_r^m \\ &= \lambda_j - \lambda_j \\ &= 0 \end{aligned}$$

since the feasibility of $\delta \nu$ implies that (8) is satisfied.

The second requirement of Bertsekas *et al.* is also satisfied since $\delta\boldsymbol{\nu}$ satisfies (9) and therefore $\Delta\nu_r = \delta\nu_r - \nu_r^m = \delta\nu_r \geq 0$ for all $r \in \mathcal{R}$ such that $\nu_r^m = 0$.

The final requirement is met by choosing the vector $\boldsymbol{\alpha} = (\alpha_r)_{r \in \mathcal{R}}$ such that $\alpha_r \in [0, 1]$ for each $r \in \mathcal{R}$ and $D(\boldsymbol{\nu}^{m+1}) < D(\boldsymbol{\nu}^m)$.

We consider two algorithms that solve Optimisation Problem 2.1 in the next two sections. The first algorithm belongs to this class. The second algorithm is similar to the algorithms in this class but the flow movement is not specified by (10).

2.3 The Kleinrock Algorithm

The version of flow deviation presented here moves the same portion α of flow to all least-cost routes. Thus $\alpha_r = \alpha$ for all $r \in \mathcal{R}$ and from (10):

$$\nu_r^{m+1} = (1 - \alpha)\nu_r^m + \alpha\delta\nu_r \quad \text{for each } r \in \mathcal{R}$$

which can be written as a vector equation $\boldsymbol{\nu}^{m+1} = (1 - \alpha)\boldsymbol{\nu}^m + \alpha\delta\boldsymbol{\nu}$. The factor α is chosen such that $D((1 - \alpha)\boldsymbol{\nu}^m + \alpha\delta\boldsymbol{\nu})$ is minimised. The result of this is that $D(\boldsymbol{\nu}^{m+1}) \leq D(\boldsymbol{\nu}^m)$. This can be seen by noting that $\alpha = 0$ yields $D(\boldsymbol{\nu}^{m+1}) = D(\boldsymbol{\nu}^m)$ and therefore the minimisation process will yield a value of α such that $D(\boldsymbol{\nu}^{m+1})$ is at most equal to $D(\boldsymbol{\nu}^m)$. The flow deviation algorithm terminates when $D(\boldsymbol{\nu}^{m+1})$ is no longer strictly less than $D(\boldsymbol{\nu}^m)$. Thus, during the execution of the algorithm $D(\boldsymbol{\nu}^{m+1}) < D(\boldsymbol{\nu}^m)$ which satisfies the third requirement mentioned at the end of section 2.2.

Link and route flows are related by (3) and therefore Optimisation Problem 2.1 can be solved with the link flows as the decision variables. The main advantage is that the number of decision variables is smaller since (in general) $R \approx N! \gg N(N - 1) \geq L$. Implicitly the problem still works with route flows. This will be explained shortly.

Optimisation Problem 2.2.

Minimise:

$$D(\boldsymbol{\gamma}) = \sum_{\ell=1}^L D_\ell(\gamma_\ell)$$

Subject to:

$$\begin{aligned} \lambda_j &= \sum_{r \in \mathcal{R}_j} \nu_r && \text{for all } j \in \mathcal{J} \\ \gamma_\ell &\geq 0 && \text{for all } \ell \in \mathcal{L}. \end{aligned}$$

The algorithm which is now discussed is a special case of the *Frank-Wolfe method* (see appendix B) and is due to Kleinrock [23].

We start with an important assumption.

Assumption 2.3. For each $\ell \in \mathcal{L}$, D_ℓ is a convex function of γ_ℓ .

D (given by (4)) is therefore also a convex function of γ_ℓ . This implies that if a feasible link flow vector exists, any link flow vector γ which minimises D yields the global minimum of D (see appendix A for a proof).

The algorithm is initialised with a feasible link flow vector ϕ^0 which is repeatedly modified until D is minimised. Kleinrock [23] provides a method of finding a feasible initial link flow vector which is discussed in chapters 3 and 4 since it depends on the link cost function D_ℓ . At each step m of the algorithm, a feasible link flow vector $\phi^m = (\phi_\ell^m)_{\ell \in \mathcal{L}}$ is constructed from the previous feasible link flow vector ϕ^{m-1} by shifting a portion α of the current flow from extremal routes onto least-cost routes and then calculating the link flow vector ϕ^m based on the resulting route flows. This is equivalent to shifting flow to and from links; thus the earlier comment that the algorithm implicitly works with routes.

A new link flow vector ϕ^m is constructed at each step m of the algorithm in the following way. Compute the cost rate vector \mathbf{c} for the previous link flow vector ϕ^{m-1} , thus

$$c_\ell = \left. \frac{\partial D}{\partial \gamma_\ell} \right|_{\gamma_\ell = \phi_\ell^{m-1}}$$

for each $\ell \in \mathcal{L}$. Note that $\mathbf{c} = \nabla D(\phi^{m-1})$. Compute the incremental network cost ΔD for the link flow vector ϕ^{m-1} where

$$\Delta D = \mathbf{c} \cdot \phi^{m-1} = \sum_{\ell=1}^L c_\ell \phi_\ell^{m-1}. \quad (11)$$

ΔD is the derivative of D at the point ϕ^{m-1} in the direction indicated by ϕ^{m-1} (see Fleming [17] for example).

We next determine whether a link flow vector ϕ^m , which yields a lower value of D than ϕ^{m-1} , can be constructed. If no such flow vector exists then ϕ^{m-1} is the optimal link flow vector and the algorithm is terminated.

Dijkstra's algorithm (see Manber [25] for example) is used to find a route r between each O-D pair j such that c_r is minimal. Let $\mathcal{R}_\mathbf{c}$ denote the set of these routes since these least-cost routes are based on the current cost rate vector \mathbf{c} . Let $\sigma = (\sigma_\ell)_{\ell \in \mathcal{L}}$ denote the least-cost link flow vector

which results when, for each O-D pair j , all the flow λ_j is sent along the least-cost route connecting j . Note that σ may not be a feasible link flow vector¹. Given that ϕ^{m-1} is a feasible link flow vector, we must ensure that $(1 - \alpha)\phi^{m-1} + \alpha\sigma$ (for the optimal value of α) is also feasible. The validity of the algorithm depends on this. The incremental network cost δD for the least-cost link flow vector σ is given by

$$\delta D = \mathbf{c} \cdot \sigma = \sum_{\ell=1}^L c_{\ell} \sigma_{\ell}$$

which is the derivative of D at the point ϕ^{m-1} in the direction indicated by σ . Combining this with equation (11) yields

$$\begin{aligned} \delta D - \Delta D &= \mathbf{c} \cdot (\sigma - \phi^{m-1}) \\ &= \nabla D(\phi^{m-1}) \cdot (\sigma - \phi^{m-1}) \end{aligned}$$

which is the derivative of D at the point ϕ^{m-1} in the direction $\sigma - \phi^{m-1}$.

If $\delta D < \Delta D$ the derivative of D in the direction $\sigma - \phi^{m-1}$ is negative and D can be decreased by assigning a portion α of the flow in this direction. This is done by moving flow to the routes in \mathcal{R}_{σ} . A line search is used to find the optimal value of $\alpha \in [0, 1]$ such that $\gamma(\alpha) = (1 - \alpha)\phi^{m-1} + \alpha\sigma$ minimises D and ϕ^m is set to $\gamma(\alpha)$. Suitable search methods include the golden section search algorithm (see Press *et al.* [27] for example) and the Fibonacci search method (see appendix C).

If $\delta D > \Delta D$ the network cost D will increase if flow is moved to the routes in \mathcal{R}_{σ} . ϕ^{m-1} represents an optimal link flow vector in this case and the algorithm is terminated.

Algorithm 2.1 (Kleinrock Flow Deviation). *We start with an initial feasible link flow vector ϕ^0 and choose a termination criterion $\epsilon_{fd} > 0$.*

1. $m \leftarrow 1$
2. Compute the current cost rate vector \mathbf{c} , where $c_{\ell} = D'_{\ell}(\phi_{\ell}^{m-1})$ for each $\ell \in \mathcal{L}$.
3. Compute the incremental cost ΔD for the current link flow vector:

$$\Delta D = \sum_{\ell=1}^L c_{\ell} \phi_{\ell}^{m-1}.$$

¹Note that the route flow vector is required to be feasible (ie satisfy definition 2.1). The requirements for a feasible link flow vector depend on the exact form of the link cost function D_{ℓ} , which in turn depends on the MPLS network being modelled. One obvious requirement is that $\gamma_{\ell} \leq C_{\ell}$.

4. Use Dijkstra's algorithm to find the least-cost routes based on the current cost rate vector \mathbf{c} . Denote the least-cost link flow vector (which results when all the flow λ_j is deviated along the least-cost routes connecting j) by σ .
5. Compute the incremental cost δD for the least-cost link flow vector:

$$\delta D = \sum_{\ell=1}^L c_{\ell} \sigma_{\ell}.$$

6. **if** $|\Delta D - \delta D| < \varepsilon_{\text{fd}}$ **then**
 stop
 else
 continue
 endif

7. Find α (with $0 \leq \alpha \leq 1$) such that the link flow vector $(1 - \alpha)\phi^{m-1} + \alpha\sigma$ minimises D and set $\phi^m = (1 - \alpha)\phi^{m-1} + \alpha\sigma$.
8. $m \leftarrow m + 1$ and **return** to step 2.

After the algorithm has terminated (at iteration $m = M$) we set

$$\gamma_{\ell} = \phi_{\ell}^{M-1} \quad \text{for each } \ell \in \mathcal{L}.$$

Note that the termination criterion $|\Delta D - \delta D| < \varepsilon_{\text{fd}}$ can be replaced by inserting the simple convergence test $|D(\nu^m) - D(\nu^{m-1})| < \varepsilon_{\text{fd}}$ between steps 7 and 8.

Computationally, the most expensive steps are 4 and 7. The standard implementation of Dijkstra's algorithm has complexity $O(N^3)$. This can be reduced to $O(N^2 \log N)$ by using heaps (see Manber [25] for example). The versions of Dijkstra's algorithm used in this thesis all make use of heaps. The line search methods used in this thesis all have lower complexities than Dijkstra's algorithm. Thus the complexity of one iteration of the Kleinrock algorithm is $O(N^2 \log N)$. The number of iterations which the algorithm requires to converge cannot be estimated. Consequently, an upper bound on the complexity of the entire algorithm cannot be given.

The discussion given here is simply a motivation for the algorithm. The algorithm is formally derived from the Frank-Wolfe method in appendix B. Note that step 4 is equivalent to solving Optimisation Problem B.7.

2.4 The Bertsekas-Gallager Algorithm

This section presents another version of the flow deviation algorithm, namely the *Bertsekas-Gallager algorithm* (see Bertsekas *et al.* [9]). This algorithm solves Optimisation Problem 2.1 by applying a gradient projection method to a Cauchy-type steepest descent method (based on Newton's method) as described in appendix D.

The least-cost routes are calculated in each iteration of the algorithm and flow is then moved from the extremal routes onto the least-cost routes. The amount of flow moved is O-D pair dependent and calculated by means of a gradient projection method. This is different from the Kleinrock algorithm (discussed in the previous section) in which the same portion of flow is moved from all extremal routes connecting all O-D pairs. Consider iteration m of the Bertsekas-Gallager algorithm and let r_j be the current least-cost route between O-D pair j . The equality constraint

$$\lambda_j = \sum_{r \in \mathcal{R}_j} \nu_r$$

can be expressed as

$$\nu_{r_j} = \lambda_j - \sum_{\substack{r \in \mathcal{R}_j \\ r \neq r_j}} \nu_r \quad (12)$$

which can be substituted into Optimisation Problem 2.1 to eliminate the equality constraints and thus yield the following problem.

Optimisation Problem 2.3.

Minimise:

$$\tilde{D}(\tilde{\nu})$$

Subject to:

$$\begin{aligned} \sum_{\substack{r \in \mathcal{R}_j \\ r \neq r_j}} \nu_r &\leq \lambda_j && \text{for all } j \in \mathcal{J} \\ \nu_r &\geq 0 && \text{for all } j \in \mathcal{J}, r \in \mathcal{R}_j \text{ and } r \neq r_j. \end{aligned}$$

$\tilde{\nu}$ consists of each route flow ν_r such that r is not a least-cost route.

The gradient projection method (described in appendix D) can now be applied to this problem. The main iteration (54) of the gradient projection method for this problem is

$$\nu_r^m = \max \left\{ 0, \nu_r^{m-1} - \left[\frac{\partial^2}{\partial \nu_r^2} \tilde{D}(\tilde{\nu}^{m-1}) \right]^{-1} \frac{\partial}{\partial \nu_r} \tilde{D}(\tilde{\nu}^{m-1}) \right\}, \quad (13)$$

where r is not a least-cost route. The above calculation is performed for each extremal route between an O-D pair j . Finally, the route flow of the least-cost route r_j is calculated as

$$\nu_{r_j}^m = \lambda_j - \sum_{\substack{r \in \mathcal{R}_j \\ r \neq r_j}} \nu_r^m.$$

The first and second partial derivatives of $\tilde{D}(\tilde{\nu})$ with respect to the route flows are calculated by using (12) and by applying the chain rule for functions of several variables to yield

$$\frac{\partial}{\partial \nu_r} \tilde{D}(\tilde{\nu}) = \frac{\partial}{\partial \nu_r} D(\nu) - \frac{\partial}{\partial \nu_{r_j}} D(\nu) \quad (14)$$

for O-D pair j and $r \neq r_j$. This can be developed further by using (6):

$$\frac{\partial}{\partial \nu_r} D(\nu) = \sum_{\ell \in r} \frac{\partial}{\partial \gamma_\ell} D(\nu) = \sum_{\ell \in r} c_\ell$$

which leads to

$$\begin{aligned} \frac{\partial}{\partial \nu_r} \tilde{D}(\tilde{\nu}) &= \sum_{\ell \in r} \frac{\partial}{\partial \gamma_\ell} D(\nu) - \sum_{\ell \in r_j} \frac{\partial}{\partial \gamma_\ell} D(\nu) \\ &= \sum_{\ell \in r} c_\ell - \sum_{\ell \in r_j} c_\ell \\ &= c_r - c_{r_j}. \end{aligned} \quad (15)$$

Define

$$c_r^2 = \frac{\partial^2 \tilde{D}(\tilde{\nu})}{(\partial \nu_r)^2}$$

and the *second derivative cost rate vector* $\mathbf{c}^2 = (c_1^2, c_2^2, \dots, c_L^2)$ as

$$c_\ell^2 = \frac{\partial^2 D}{\partial (\gamma_\ell)^2} \quad \text{for each } \ell \in \mathcal{L}.$$

(15) and the same argument used in the derivation of (6) give

$$\begin{aligned}
 c_r^2 &= \frac{\partial}{\partial \nu_r} \left[\sum_{\ell \in r} \frac{\partial}{\partial \gamma_\ell} D(\nu) - \sum_{\ell \in r_j} \frac{\partial}{\partial \gamma_\ell} D(\nu) \right] \\
 &= \sum_{\ell \in r} \frac{\partial^2}{\partial (\gamma_\ell)^2} D(\nu) - \sum_{\ell \in r_j} \frac{\partial^2}{\partial (\gamma_\ell)^2} D(\nu) \\
 &= \sum_{\ell \in \mathcal{L}_r} \frac{\partial^2}{\partial (\gamma_\ell)^2} D(\nu) \\
 &= \sum_{\ell \in \mathcal{L}_r} c_\ell^2,
 \end{aligned}$$

where \mathcal{L}_r is the set of all links in r or r_j but not in both:

$$\mathcal{L}_r = \{\ell \in \mathcal{L} \mid \ell \in r \text{ or } \ell \in r_j\} \setminus \{\ell \in \mathcal{L} \mid \ell \in r \text{ and } \ell \in r_j\}.$$

Consequently the iterative step (13) of the gradient projection method becomes

$$\nu_r^m = \max \left\{ 0, \nu_r^{m-1} - \frac{c_r - c_{r_j}}{c_r^2} \right\}.$$

Algorithm 2.2 (Bertsekas-Gallager Flow Deviation). *We start with an initial feasible link flow vector ϕ^0 and choose a termination criterion $\varepsilon_{fd} > 0$ and a step-size $\eta > 0$.*

1. $m \leftarrow 1$
2. Compute the current cost rate vector \mathbf{c} , where $c_\ell = D'_\ell(\phi_\ell^{m-1})$ for each $\ell \in \mathcal{L}$.
3. Compute the incremental cost ΔD for the current link flow vector:

$$\Delta D = \sum_{\ell=1}^L c_\ell \phi_\ell^{m-1}.$$

4. Use Dijkstra's algorithm to find the least-cost routes based on the current cost rate vector \mathbf{c} . Denote the least-cost link flow vector (which results when all the flow λ_j is deviated along the least-cost routes connecting j) by σ .
5. Compute the incremental cost δD for the least-cost link flow vector:

$$\delta D = \sum_{\ell=1}^L c_\ell \sigma_\ell.$$

6. **if** $|\Delta D - \delta D| < \varepsilon_{fd}$ **then**

stop

else

continue

endif

7. Set $\phi^m \leftarrow \phi^{m-1}$ and perform the following procedure for each O-D pair j .

(a) Compute the cost rate vector \mathbf{c} , where $c_\ell = D'_\ell(\phi_\ell^m)$ for each $\ell \in \mathcal{L}$ and the second derivative cost rate vector \mathbf{c}^2 , where $c_\ell^2 = D''_\ell(\phi_\ell^m)$ for each $\ell \in \mathcal{L}$.

(b) Let r_j be the least-cost route between O-D pair j . Then for each route $r \in \mathcal{R}_j$, $r \neq r_j$ set

$$\omega_r = \eta \frac{c_r - c_{r_j}}{c_r^2}$$

and set

$$\nu_r^m = \max \{0, \nu_r^{m-1} - \omega_r\}.$$

(c) Calculate the flow along the least-cost route connecting O-D pair j :

$$\nu_{r_j}^m = \lambda_j - \sum_{\substack{r \in \mathcal{R}_j \\ r \neq r_j}} \nu_r^m.$$

(d) Update the link flow vector ϕ^m , where

$$\phi_\ell^m = \sum_{r \in \mathcal{A}_\ell} \nu_r^m \quad \text{for each } \ell \in \mathcal{L}.$$

Decrement the step-size η in an appropriate way.

8. $m \leftarrow m + 1$ and **return** to step 2.

After the algorithm has terminated (in iteration $m = M$) we set

$$\gamma_\ell = \phi_\ell^{M-1} \quad \text{for each } \ell \in \mathcal{L}.$$

Note that the termination criterion $|\Delta D - \delta D| < \varepsilon_{fd}$ can be replaced by inserting the simple convergence test $|D(\nu^m) - D(\nu^{m-1})| < \varepsilon_{fd}$ between steps 7 and 8.

The link flow vector ϕ^m is updated (in step 7) after the LSP flows $\nu_r \in \mathcal{R}_j$ have been changed for each O-D pair j . Therefore ϕ^m is only fixed at the completion of iteration m . This implies that the cost rate vectors \mathbf{c} and \mathbf{c}^2 have to be recalculated for each O-D pair.

A step-size η is introduced in the iterative step. Bertsekas *et al.* [9] suggest an initial step-size of $\eta = 1$. They also mention the possibility of keeping η fixed as opposed to several decrement procedures.

Computationally, the most expensive step is the calculation of the shortest paths. The version of Dijkstra's algorithm used in this thesis has complexity $O(N^2 \log N)$ (see the explanation for the Kleinrock algorithm). Thus one iteration of the Bertsekas-Gallager algorithm has complexity $O(N^2 \log N)$.

2.5 Applying Flow Deviation to MPLS Networks

The next two chapters adapt the flow deviation algorithms to optimise the QoS offered by specific MPLS networks. Chapter 3 considers an MPLS network model in which the major packet delays occur in the nodes (LSRs). This model accommodates different service classes. Chapter 4 applies flow deviation to MPLS network models in which the limited link bandwidths cause the major delays. This model is only applicable to traffic belonging to a single service class.

Chapter 3

Minimising Nodal Delays in MPLS Networks

This chapter applies the flow deviation algorithm from chapter 2 to a specific MPLS network model. The cost function (which was not specified in chapter 2) is the expected packet delay in the network. An important characteristic of the network model considered here is that the links are so well-dimensioned that their capacities can be regarded as infinite. Consequently all delays in the network occur in the nodes.

Section 3.1 introduces a mathematical model of the MPLS network considered in this chapter. An expression for the expected packet delay is derived in section 3.2. The queueing mechanism in the label switching routers is discussed in section 3.3. Section 3.4 presents an algorithm for constructing optimal label switched paths. This algorithm is applied to some test networks in section 3.5.

3.1 The Model

Consider an MPLS network with N nodes which carries K service classes¹. Each node corresponds to a *label switching router* (LSR) which consists of a queue and a service facility. Let $\mathcal{N} = \{1, 2, \dots, N\}$ denote the set of nodes and let $\mathcal{K} = \{1, 2, \dots, K\}$ denote the set of service classes. The length of a class k packet² is an exponential random variable with mean $1/\mu_k$. The random variables corresponding to the packet lengths of the different service classes are independent. These random variables also represent the rates at which the packets are served by the LSRs.

Let \mathcal{J} denote the set of all O-D pairs. The O-D pairs (o, d) and (d, o) are distinct and $J = |\mathcal{J}| = N(N - 1)$. Some O-D pairs are connected by physical links and the links are identified by the

¹The words *service class* and *traffic class* are used interchangeably.

²A packet always refers to an IP packet in this discussion.

numbers of the O-D pairs which they connect. Each link is bi-directional and is identified by two link numbers. Let \mathcal{L} denote the set of links. The function $\mathcal{L} : \mathcal{N} \times \mathcal{N} \mapsto \mathcal{J}$ is defined as $\mathcal{L}(o, d) = j$ if O-D pair (o, d) is numbered j .

Let C_ℓ denote the transmission capacity (zero if the physical link does not exist) of the bi-directional link $o - d$ for each O-D pair (o, d) such that $\mathcal{L}(o, d) = \ell$. The bi-directionality of the links implies that $C_m = C_\ell$ where $\mathcal{L}(d, o) = m$. The link capacities C_ℓ are sufficiently large so that *link delays* can be ignored. Propagation delays are assumed to be negligible. This “infinite bandwidth” model implies that only *nodal delays* are considered. The set \mathcal{L} can be expressed as

$$\mathcal{L} = \{j \in \mathcal{J} \mid C_j > 0\} \subseteq \mathcal{J}.$$

Let λ_{jk} denote the Poisson arrival rate of class k packets to O-D pair j . Let λ be the total packet arrival rate to the network, therefore

$$\lambda = \sum_{j=1}^J \sum_{k=1}^K \lambda_{jk}.$$

A *label switched path* (LSP) $r = \langle o, d \rangle$ connecting nodes o and d is a sequence of physical links $o - o_1, o_1 - o_2, \dots, o_m - d$. The expression $n \in r$ indicates that node n is traversed by LSP r . Note that (o, d) denotes the O-D pair, $o - d$ denotes the physical link and $\langle o, d \rangle$ denotes an LSP connecting nodes o and d . Let \mathcal{R}_j denote the set of LSPs (including the direct link if it exists) between O-D pair j . Figure 3.1 is an example of an LSP set $\mathcal{R}_j = \{r_1, r_2\}$, where $\mathcal{L}(o_1, d_1) = j$. Let \mathcal{R} be the set of all LSPs in the network, ie

$$\mathcal{R} = \bigcup_{j \in \mathcal{J}} \mathcal{R}_j.$$

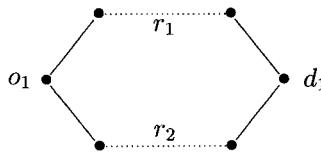
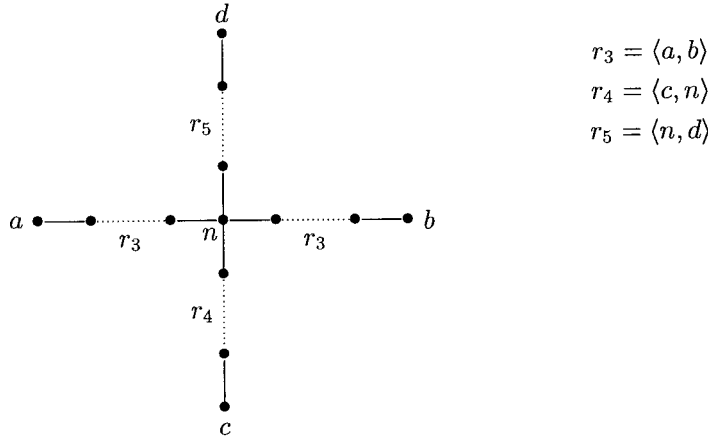


Figure 3.1: The Set \mathcal{R}_j

Let \mathcal{B}_n denote the set of LSPs passing through node n . \mathcal{B}_n includes LSPs that originate or terminate at n . Each node n can be a *transit node* in an LSP and/or an *ingress node* and/or an

egress node. Similarly a packet entering a node can be classified as a *transit*, *ingress* or *egress* packet. Since each node n corresponds to an LSR (possibly) connected to a subnetwork, ingress packets that originate in the subnetwork connected to n also enter n and have to be switched in the same manner as transit packets. The only difference between the *label swapping* performed on ingress packets and other packets is that the ingress packets are not labeled when they enter n . The same holds for egress packets destined for the subnetwork to which n is connected. The only difference between the label swapping performed on egress packets and other packets is that the labels are removed from egress packets and the packets are passed on to the appropriate subnetwork device instead of being forwarded to another LSP. Figure 3.2 illustrates an instance of the set $\mathcal{B}_n = \{r_3, r_4, r_5\}$ where node n is a transit node for LSP r_3 , an egress node for LSP r_4 and an ingress node for LSP r_5 .


 Figure 3.2: The Set \mathcal{B}_n

3.2 The Expected Packet Delay in MPLS Networks

Let λ_j be the total packet arrival rate to O-D pair j :

$$\lambda_j = \sum_{k=1}^K \lambda_{jk}.$$

All packets offered to O-D pair j are switched along LSPs in \mathcal{R}_j . All LSPs are *service integrated* in this network model – packets are not routed based on service class, rather all packets belonging to all service classes are combined and the combined packet stream is switched along the appropriate LSPs. The flow along each LSP consists of packets belonging to any service class but to the same *forwarding equivalence class* (FEC). Let s_{rj} denote the portion of the packets offered to O-D pair j which travel along LSP r . Therefore s_{rj} is the portion of the packets offered to O-D pair j that

belong to the FEC to which LSP r is assigned. Note that if $r \notin \mathcal{R}_j$ then $s_{rj} = 0$. Let ν_{rk} denote the arrival rate of class k packets to LSP r , therefore $\nu_{rk} = s_{rj}\lambda_{jk}$ for all $j \in \mathcal{J}$, $k \in \mathcal{K}$ and $r \in \mathcal{R}_j$. Since all packets offered to O-D pair j are switched along LSPs in \mathcal{R}_j ,

$$\lambda_{jk} = \sum_{r \in \mathcal{R}_j} \nu_{rk} \quad (16)$$

$$\sum_{r \in \mathcal{R}_j} s_{rj} = 1 \quad (17)$$

for all $j \in \mathcal{J}$ and $k \in \mathcal{K}$.

Let γ_{nk} denote the rate at which class k packets arrive at node n and let γ_n denote the total packet arrival rate at node n , then

$$\gamma_n = \sum_{k=1}^K \gamma_{nk} \quad \text{for all } n \in \mathcal{N}. \quad (18)$$

The relation between γ_{nk} and ν_{rk} is

$$\gamma_{nk} = \sum_{r \in \mathcal{B}_n} \nu_{rk} \quad \text{for all } n \in \mathcal{N} \text{ and } k \in \mathcal{K}. \quad (19)$$

Let the random variable D represent the delay a packet experiences in the network and let $T = \mathbf{E}[D]$. Let the random variable d_{jk} represent the delay a class k packet offered to O-D pair j experiences and let $t_{jk} = \mathbf{E}[d_{jk}]$, then

$$\begin{aligned} T &= \sum_{j=1}^J \frac{\lambda_j}{\lambda} \sum_{k=1}^K \frac{\lambda_{jk}}{\sum_{k=1}^K \lambda_{jk}} t_{jk} \\ &= \sum_{j=1}^J \frac{\lambda_j}{\lambda} \sum_{k=1}^K \frac{\lambda_{jk}}{\lambda_j} t_{jk} \\ &= \sum_{j=1}^J \frac{1}{\lambda} \sum_{k=1}^K \lambda_{jk} t_{jk} \\ &= \sum_{j=1}^J \sum_{k=1}^K \frac{\lambda_{jk}}{\lambda} t_{jk}, \end{aligned} \quad (20)$$

which is a decomposition of the expected packet delay in terms of O-D pair.

We now define the random variable D_{nk} as the *nodal delay*, which comprises the time that a class k packet spends waiting in the queue and being served at node n . Since the link capacities C_ℓ

are assumed to be “sufficient”, we only consider nodal delays and therefore the expected values $T_{nk} = \mathbf{E}[D_{nk}]$ and t_{jk} are related by

$$t_{jk} = \sum_{r \in \mathcal{R}_j} s_{rj} \sum_{n \in r} T_{nk}$$

and therefore (using (20)) the expected packet delay T can be expressed in terms of the expected nodal delays:

$$\begin{aligned} T &= \sum_{j=1}^J \sum_{k=1}^K \frac{\lambda_{jk}}{\lambda} t_{jk} \\ &= \sum_{j=1}^J \sum_{k=1}^K \frac{\lambda_{jk}}{\lambda} \sum_{r \in \mathcal{R}_j} s_{rj} \sum_{n \in r} T_{nk} \\ &= \sum_{j=1}^J \sum_{k=1}^K \frac{1}{\lambda} \sum_{r \in \mathcal{R}_j} s_{rj} \lambda_{jk} \sum_{n \in r} T_{nk} \\ &= \sum_{k=1}^K \sum_{j=1}^J \frac{1}{\lambda} \sum_{r \in \mathcal{R}_j} \nu_{rk} \sum_{n \in r} T_{nk} \\ &= \sum_{k=1}^K \sum_{r \in \mathcal{R}} \frac{1}{\lambda} \nu_{rk} \sum_{n \in r} T_{nk} \\ &= \sum_{k=1}^K \sum_{n=1}^N \frac{1}{\lambda} T_{nk} \sum_{r \in \mathcal{B}_n} \nu_{rk} \\ &= \sum_{n=1}^N \sum_{k=1}^K \frac{\gamma_{nk}}{\lambda} T_{nk}, \end{aligned} \tag{21}$$

where equation (19) has been used.

3.3 The LSR's Queueing Discipline

The form of equation (21) depends on the expected nodal delay T_{nk} , which in turn depends on the LSR's queueing discipline. Whereas packet flows on LSPs are service integrated, *service separation* is enforced in each LSR. Each service class is provided with a dedicated server and an unbounded waiting line. All the class k packets are extracted from the incoming packet stream and sent to the queue serving class k packets. This is modelled as an $M/M/1$ queue with arrival rate γ_{nk} and service rate μ_k and therefore the expected nodal delay T_{nk} is the expected waiting time

$$T_{nk} = \frac{1}{\mu_k - \gamma_{nk}}. \tag{22}$$

3.4 Finding Optimal LSPs

This section presents an algorithm for finding optimal LSPs for all O-D pairs in an MPLS network. The optimal LSPs are expressed in terms of the set \mathcal{R} of LSPs and the factors s_{rj} which specify along which LSPs the packets are to be switched in order to minimise the expected packet delay. The algorithm is based on the Kleinrock flow deviation algorithm (see chapter 2). Chapter 2 discusses this algorithm in the context of a general MPLS network with a *link cost function*. We now consider its application to a specific MPLS network with a *nodal cost function*. We give a short motivation³ for the node-based algorithm similar to the motivation given for the link-based algorithm in chapter 2. The problem of finding optimal LSPs is stated in terms of the nodal arrival rates γ_{nk} instead of the LSP arrival rates ν_{rk} . This is similar to the approach in chapter 2 where the initial optimisation problem (Optimisation Problem 2.1) with route flows as decision variables is transformed into an optimisation problem (Optimisation Problem 2.2) with link flows as decision variables.

The problem of finding an optimal packet *flow vector* $\gamma = (\gamma_{nk})_{n \in \mathcal{N}, k \in \mathcal{K}}$ can be stated as the following non-linear optimisation problem.

Optimisation Problem 3.1.

Minimise:

$$T(\gamma) = \frac{1}{\lambda} \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} T_{nk}$$

Subject to:

$$\begin{aligned} \lambda_{jk} &= \sum_{r \in \mathcal{R}_j} \nu_{rk} \quad \text{for all } j \in \mathcal{J} \text{ and } k \in \mathcal{K} \\ \gamma_{nk} &= \sum_{r \in \mathcal{B}_n} \nu_{rk} \quad \text{for all } n \in \mathcal{N} \text{ and } k \in \mathcal{K}. \end{aligned}$$

An important concept used in the flow deviation algorithm is that of a *feasible* flow vector.

Definition 3.1 (Feasible Flow Vector). *A flow vector γ is feasible if the arrival rate $\gamma_{nk} < \mu_k$ for each node $n \in \mathcal{N}$ and each service class $k \in \mathcal{K}$.*

In order to solve the optimisation problem for a given network we require a feasible flow vector to exist. This is a necessary condition for the $M/M/1$ queues in the LSRs to reach statistical equilibrium (see Cooper [12] or Wolff [34], for example) and therefore for (22) to be valid.

Substituting (22) into (21) yields

³The arguments presented in this chapter provide a motivation, not a proof. The algorithm is based on the Frank-Wolfe method which is formally derived in appendix B.

$$\begin{aligned}
 T &= \frac{1}{\lambda} \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} T_{nk} \\
 &= \frac{1}{\lambda} \sum_{n=1}^N \sum_{k=1}^K \frac{\gamma_{nk}}{(\mu_k - \gamma_{nk})}
 \end{aligned} \tag{23}$$

and therefore

$$\begin{aligned}
 \frac{\partial T}{\partial \gamma_{nk}} &= \frac{\mu_k}{\lambda(\mu_k - \gamma_{nk})^2} \\
 \frac{\partial^2 T}{\partial \gamma_{nk}^2} &= \frac{2\mu_k}{\lambda(\mu_k - \gamma_{nk})^3}
 \end{aligned} \tag{24}$$

for all $n \in \mathcal{N}$ and $k \in \mathcal{K}$. These equations imply that the first two partial derivatives of T with respect to γ_{nk} are non-negative for all $n \in \mathcal{N}$ and $k \in \mathcal{K}$. Therefore T is a convex function of γ_{nk} and any feasible flow vector γ which minimises T yields the global minimum of T (see appendix A).

The algorithm is initialised with a feasible flow vector ϕ^0 which is repeatedly modified until T is minimised. At each step m of the algorithm a feasible flow vector $\phi^m = (\phi_{nk}^m)_{n \in \mathcal{N}, k \in \mathcal{K}}$ is constructed from the previous feasible flow vector ϕ^{m-1} by shifting a portion α of the current flow on the existing LSPs or from existing LSPs on to newly-found LSPs.

Define a *cost rate vector* $\mathbf{c} = (c_{nk})_{n \in \mathcal{N}, k \in \mathcal{K}}$ where $c_{nk} = \partial T / \partial \gamma_{nk}$. Thus if the flow along LSP r is increased such that the class k packet arrival rate to node n increases by a small amount σ_{nk} , then (to first order in σ_{nk}) T increases by $\delta T = c_{nk} \sigma_{nk}$. The vector \mathbf{c} is the gradient vector of T at the point γ and is denoted by $\nabla T(\gamma)$.

At each step m of the algorithm a new flow vector ϕ^m is constructed as follows. Compute the cost rate vector \mathbf{c} for the previous flow vector ϕ^{m-1} , thus

$$c_{nk} = \left. \frac{\partial T}{\partial \gamma_{nk}} \right|_{\gamma_{nk} = \phi_{nk}^{m-1}}$$

for each $n \in \mathcal{N}$ and $k \in \mathcal{K}$. Compute the *incremental delay* ΔT for the flow vector ϕ^{m-1} , where

$$\Delta T = \mathbf{c} \cdot \phi^{m-1} = \sum_{n=1}^N \sum_{k=1}^K c_{nk} \phi_{nk}^{m-1}. \tag{25}$$

ΔT is the derivative of T at the point ϕ^{m-1} in the direction indicated by ϕ^{m-1} (see Fleming [17], for example).

We next determine whether a flow vector ϕ^m which yields a lower value of T than ϕ^{m-1} can be constructed. If no such flow vector exists then ϕ^{m-1} is the optimal flow vector and the algorithm is terminated. Let c_r denote the cost of LSP $r \in \mathcal{R}_j$:

$$c_r = \frac{1}{\lambda_j} \sum_{n \in r} \sum_{k=1}^K \lambda_{jk} c_{nk}.$$

Dijkstra's algorithm (see Manber [25] or Tucker [31], for example) is used to find an LSP r between each O-D pair j such that c_r is minimal. Let \mathcal{R}_c denote the set of these LSPs since these least-cost LSPs are based on the current cost rate vector \mathbf{c} . Let $\sigma = (\sigma_{nk})_{n \in \mathcal{N}, k \in \mathcal{K}}$ denote the least-cost flow vector which results when, for each O-D pair j , all the flow λ_j is sent along the least-cost LSP connecting j . Note that σ may not be a feasible flow vector – see below.

The incremental delay δT for the least-cost flow vector σ is given by

$$\delta T = \mathbf{c} \cdot \sigma = \sum_{n=1}^N \sum_{k=1}^K c_{nk} \sigma_{nk}$$

which is the derivative of T at the point ϕ^{m-1} in the direction indicated by σ . Combining this with equation (25) yields

$$\begin{aligned} \delta T - \Delta T &= \mathbf{c} \cdot (\sigma - \phi^{m-1}) \\ &= \nabla T(\phi^{m-1}) \cdot (\sigma - \phi^{m-1}), \end{aligned}$$

which is the derivative of T at the point ϕ^{m-1} in the direction $\sigma - \phi^{m-1}$.

If $\delta T < \Delta T$ the derivative of T in the direction $\sigma - \phi^{m-1}$ is negative and T can be decreased by assigning a portion α of the flow in this direction. This is accomplished by moving flow to the LSPs in \mathcal{R}_c . A *line search* is used to find the optimal value of $\alpha \in [0, 1]$ such that $\gamma(\alpha) = (1 - \alpha)\phi^{m-1} + \alpha\sigma$ minimises T and ϕ^m is set to $\gamma(\alpha)$. Suitable line search methods include the golden section search algorithm (see Press *et al.* [27], for example) and the Fibonacci search method (see appendix C).

If $\delta T > \Delta T$ the expected delay T will increase if flow is moved to the LSPs in \mathcal{R}_c . Therefore ϕ^{m-1} represents an optimal flow vector and the algorithm is terminated.

Given that ϕ^{m-1} is a feasible flow vector, we must ensure that $(1 - \alpha)\phi^{m-1} + \alpha\sigma$ (for the optimal value of α) is also feasible. The validity of the algorithm depends on this. If σ is feasible then $(1 - \alpha)\phi^{m-1} + \alpha\sigma$ is feasible since

$$(1 - \alpha)\phi_{nk}^{m-1} + \alpha\sigma_{nk} < (1 - \alpha)\mu_k + \alpha\mu_k = \mu_k$$

for all $n \in \mathcal{N}$ and $k \in \mathcal{K}$. However, there is no guarantee that σ is feasible. Consider the situation where σ is not feasible. Suppose an $n \in \mathcal{N}$ and a $k \in \mathcal{K}$ exist such that

$$\gamma_{nk} = (1 - \alpha)\phi_{nk}^{m-1} + \alpha\sigma_{nk} \geq \mu_k. \quad (26)$$

It follows from (23) that

$$\lim_{\gamma_{nk} \rightarrow \mu_k^-} \partial T / \partial \gamma_{nk} = \infty$$

which implies that if $\gamma_{nk} \rightarrow \mu_k^-$ for any $n \in \mathcal{N}$ and $k \in \mathcal{K}$, T grows without bound. The left-hand limit is taken since T (and consequently $\partial T / \partial \gamma_{nk}$) is not defined for $\gamma_{nk} > \mu_k$.

Therefore a value of α satisfying the equality in equation (26) is never selected since the line search attempts to minimise T . It is important to check that the line search method does not select a value of α which satisfies the strict inequality in equation (26). This can happen since $\partial T / \partial \gamma_{nk}$ is finite for $\gamma_{nk} > \mu_k$ despite the fact that $\partial T / \partial \gamma_{nk}$ does not “physically” exist for $\gamma_{nk} > \mu_k$.

The optimisation algorithm can be stated as follows.

Algorithm 3.1 (Finding an optimal γ). *The algorithm starts with an initial feasible flow vector ϕ^0 and a termination criterion $\varepsilon > 0$.*

1. $m \leftarrow 1$.
2. Compute the current cost rate vector \mathbf{c} , where

$$c_{nk} = \left. \frac{\partial T}{\partial \gamma_{nk}} \right|_{\gamma_{nk} = \phi_{nk}^{m-1}}$$

for each $n \in \mathcal{N}$ and $k \in \mathcal{K}$.

3. Compute the incremental delay ΔT for the previous flow vector:

$$\Delta T = \sum_{n=1}^N \sum_{k=1}^K c_{nk} \phi_{nk}^{m-1}.$$

4. Use Dijkstra's algorithm to find the least-cost LSPs based on the current cost rate vector \mathbf{c} . Let σ denote the least-cost flow vector which results when, for each $j \in \mathcal{J}$, all the flow λ_j is sent along the least-cost LSP connecting j .
5. Compute the incremental delay δT for the least-cost flow vector:

$$\delta T = \sum_{n=1}^N \sum_{k=1}^K c_{nk} \sigma_{nk}.$$

6. **if** $\Delta T - \delta T < \varepsilon$ **then**
 go to step 9
 else
 continue
 endif
7. **Line search:** Find α ($0 \leq \alpha \leq 1$) such that the flow vector $\gamma(\alpha) = (1 - \alpha)\phi^{m-1} + \alpha\sigma$ minimises T . Set $\phi^m = \gamma(\alpha)$.
8. $m \leftarrow m + 1$ and **return** to step 2.
9. Set $\gamma = \phi^{m-1}$.

3.4.1 Initial Feasible Flow Vector

The above algorithm relies on the existence of a feasible initial flow vector ϕ^0 . Constructing a ϕ^0 or even determining whether a ϕ^0 exists can be a non-trivial task for large networks. Kleinrock [23] presents an algorithm which performs this task.

The algorithm starts by computing the cost rate vector \mathbf{c} when no packets are switched through the network, thus

$$c_{nk} = \left. \frac{\partial T}{\partial \gamma_{nk}} \right|_{\gamma_{nk}=0} = \frac{1}{\lambda \mu_k}$$

for each $n \in \mathcal{N}$ and $k \in \mathcal{K}$. Next it uses Dijkstra's algorithm to find the least-cost LSPs based on this cost rate vector and constructs the flow vector φ^0 which results when all packets are switched along these LSPs.

The algorithm's main loop starts with the flow vector φ^0 (which might be infeasible) and attempts to adapt it into a feasible flow vector. The algorithm will detect if this is impossible.

At each step m the algorithm attempts to route a portion h_{m-1} of the total flow λ through the network. The algorithm determines whether the flow vector φ^{m-1} (which corresponds to a total

flow of $h_{m-1}\lambda$ being offered to the network) is feasible. This is done by computing the *maximum LSR load* β , where

$$\beta = \max \left\{ \frac{\varphi_{nk}^{m-1}}{\mu_k} \mid n \in \mathcal{N} \text{ and } k \in \mathcal{K} \right\}.$$

If $\beta < h_{m-1}$ then $\varphi_{nk}^{m-1}/h_{m-1} < \mu_k$ for all $n \in \mathcal{N}$ and $k \in \mathcal{K}$ and φ^{m-1}/h_{m-1} is a feasible flow vector corresponding to the total flow λ being routed through the network. Therefore the algorithm terminates.

If $\beta \geq h_{m-1}$ then φ^{m-1}/h_{m-1} is not a feasible flow vector. The algorithm now calculates $h_m = h_{m-1}(1 - \varepsilon(1 - \beta))/\beta$. Since $0 < \varepsilon < 1$ it follows that $h_m < h_{m-1}$. A flow vector $\xi = (\xi_{nk})_{n \in \mathcal{N}, k \in \mathcal{K}}$ which corresponds to a total flow $h_m\lambda$ being routed through the network is now constructed:

$$\xi_{nk} = \frac{h_m}{h_{m-1}} \varphi_{nk}^{m-1} \quad (27)$$

for all $n \in \mathcal{N}$ and $k \in \mathcal{K}$. Based on this flow vector, the current cost rate vector \mathbf{c} is computed as

$$c_{nk} = \left. \frac{\partial T}{\partial \gamma_{nk}} \right|_{\gamma_{nk} = \xi_{nk}}$$

for each $n \in \mathcal{N}$ and $k \in \mathcal{K}$. Dijkstra's algorithm is applied to find the least-cost LSPs. Let σ denote the flow vector which results when, for each $j \in \mathcal{J}$, the flow $h_m\lambda_j$ is sent along the least-cost LSP connecting O-D pair j .

A portion α of the flow is now shifted from the current LSPs to the least-cost LSPs. The value of α is such that the flow vector $\gamma(\alpha) = (1 - \alpha)\xi + \alpha\sigma$ minimises T and φ^m is set to $\gamma(\alpha)$.

However, it is possible that this technique does not find a feasible flow. In order to determine whether a feasible flow vector can be found, the algorithm tests the following infeasibility conditions:

$$\begin{aligned} |\nabla T(\xi) \cdot (\sigma - \xi)| &= |\mathbf{c} \cdot (\sigma - \xi)| \\ &= \left| \sum_{n=1}^N \sum_{k=1}^K c_{nk} (\sigma_{nk} - \xi_{nk}) \right| \\ &< \varepsilon_1 \end{aligned} \quad (28)$$

and

$$|h_m - h_{m-1}| < \varepsilon_2, \quad (29)$$

where ε_1 and ε_2 are small positive real numbers. If both conditions are satisfied then no feasible flow vector exists.

If equation (28) is satisfied, the derivative of T at the point ξ in the direction $\sigma - \xi$ is zero. The vector $\sigma - \xi$ indicates the direction of largest decrease in T from the point ξ since σ is based on the least-cost LSPs. Therefore if T does not decrease along the direction of largest decrease at point ξ then T has a minimum at ξ .

If equation (29) is true, then it follows from equation (27) that

$$\xi = \frac{h_m}{h_{m-1}} \varphi^{m-1} \approx \varphi^{m-1}.$$

Conditions (28) and (29) together imply that $\xi = \varphi^{m-1}$ yields the minimum of T and therefore the line search yields $\alpha = 0$ and thus $\varphi^m = \varphi^{m-1}$. However, it has already been determined in iteration $m - 1$ that φ^{m-1}/h_{m-1} is not feasible and therefore φ^m/h_m is not feasible either. Consequently if the conditions (28) and (29) are met the algorithm is unable to find a feasible flow.

The algorithm can be stated as follows.

Algorithm 3.2 (Constructing a feasible ϕ^0). Set $h_0 \leftarrow 1$ and initialise the cost rate vector \mathbf{c}

$$c_{nk} = 1/\lambda\mu_k \quad \text{for each } n \in \mathcal{N} \text{ and } k \in \mathcal{K}.$$

Use Dijkstra's algorithm to find the least-cost LSPs based on this cost rate vector. Let $\varphi^0 = (\varphi_{nk}^0)_{n \in \mathcal{N}, k \in \mathcal{K}}$. Denote the flow vector which results when, for each $j \in \mathcal{J}$, all flow λ_j is sent along the least-cost LSP connecting j .

Choose a feasible starting flow precision criterion ε (with $0 < \varepsilon < 1$), and two infeasibility criteria $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$.

1. $m \leftarrow 1$
2. Calculate β , where

$$\beta = \max \left\{ \frac{\varphi_{nk}^{m-1}}{\mu_k} \mid n \in \mathcal{N} \text{ and } k \in \mathcal{K} \right\}.$$

3. **if** $\beta < h_{m-1}$ **then**
 $\phi^0 \leftarrow \varphi^{m-1}/h_{m-1}$
 stop
 else
 continue
 endif

4. $h_m \leftarrow h_{m-1}(1 - \varepsilon(1 - \beta))/\beta$

5. Let ξ denote a flow vector that routes a total flow $h_m\lambda < \lambda$ through the network, where

$$\xi_{nk} = \frac{h_m}{h_{m-1}} \varphi_{nk}^{m-1} \quad \text{for all } n \in \mathcal{N} \text{ and } k \in \mathcal{K}.$$

6. Compute the current cost rate vector \mathbf{c} , where

$$c_{nk} = \left. \frac{\partial T}{\partial \gamma_{nk}} \right|_{\gamma_{nk} = \xi_{nk}}$$

for each $n \in \mathcal{N}$ and $k \in \mathcal{K}$.

7. Use Dijkstra's algorithm to find the least-cost LSPs based on the current cost rate vector \mathbf{c} . Denote the flow vector which results when for each O-D pair j the flow $h_m\lambda_j$ is deviated along the least-cost LSP connecting j by σ .

8. If the following conditions are both met no feasible flow vector can be found and the algorithm is terminated.

$$\left| \sum_{n=1}^N \sum_{k=1}^K c_{nk} (\sigma_{nk} - \xi_{nk}) \right| < \varepsilon_1$$

$$|h_m - h_{m-1}| < \varepsilon_2.$$

9. Find α (with $0 \leq \alpha \leq 1$) such that the flow vector $\gamma(\alpha) = (1 - \alpha)\xi + \alpha\sigma$ minimises T and set $\varphi^m = \gamma(\alpha)$.

10. $m \leftarrow m + 1$ and **return** to step 2.

3.5 Some Applications

The algorithm in section 3.4 has been implemented using the golden section search method for the line search. The algorithm is applied to some test networks in this section.

3.5.1 Two Small Networks

The first network is the small single-class network (see Davie *et al.* [13]) shown in figure 3.3. Let the packet arrival rates to O-D pairs (1,6) and (2,6) be 0.4 for each O-D pair. The service rate $\mu_1 = 1$. No packets are offered to any other O-D pair. Since this network is symmetric, the optimal routing pattern results in $\gamma_{41} = \gamma_{51} = 0.4$ in order to give a total expected delay of $T = 13\frac{1}{3}$ time units. Packets travelling between O-D pair (1,6) use the LSP 1 – 3 – 4 – 6 and packets travelling between O-D pair (2,6) use the LSP 2 – 3 – 5 – 6. In order to achieve this optimal routing LSR 3 needs to route packets from LSR 1 and LSR 2 differently. This is done based on the packet labels. Davie *et al.* [13] mention that this is difficult to achieve if node 3 is replaced by a conventional router which forwards packets based only on IP destination addresses.

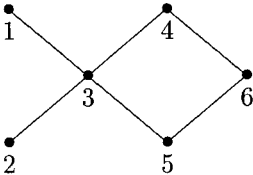


Figure 3.3: A 6 node network

The second test network is shown in figure 3.4. This network carries two service classes $\mathcal{K} = \{1, 2\}$. The arrival and service rates are given in table 3.1.

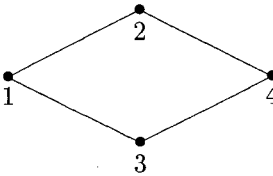


Figure 3.4: A 4 node network

	O-D pair	$k = 1$	$k = 2$
λ_{jk}	(1, 2)	0	0.5
	(1, 3)	0.5	0
	(1, 4)	0.3	0.3
	(2, 4)	0.3	0.3
	(3, 4)	0.3	0.3
μ_k		1	1

Table 3.1: The arrival and service rates for the 4 node network

This network is symmetric except for the arrival rates to O-D pairs (1,2) and (1,3). Therefore packets are routed optimally if half the packets offered to O-D pair (1,4) are sent along the LSP 1 – 2 – 4 and the other half along the LSP 1 – 3 – 4 which yields an expected delay of $T = 23.44$

time units. The packets offered to the directly connected O-D pairs are sent along the physical links connecting the O-D pairs. Due to the multi-class nature of this network there exists no feasible flow vector γ which corresponds to sending all the packets offered to O-D pair (1, 4) along a single LSP. If all these packets are sent along LSP 1 – 2 – 4 then $\gamma_{21} = 0.6 < 1 = \mu_1$ but $\gamma_{22} = 1.1 > 1 = \mu_2$ and if all these packets are sent along LSP 1 – 3 – 4 then $\gamma_{32} = 0.6 < 1 = \mu_2$ but $\gamma_{31} = 1.1 > 1 = \mu_1$.

3.5.2 A Larger Network

Consider the MPLS network in figure 3.5 which is a fictitious network connecting South Africa’s nine provincial capitals. Each line indicates a bi-directional link with sufficient capacity such that link delays can be ignored.

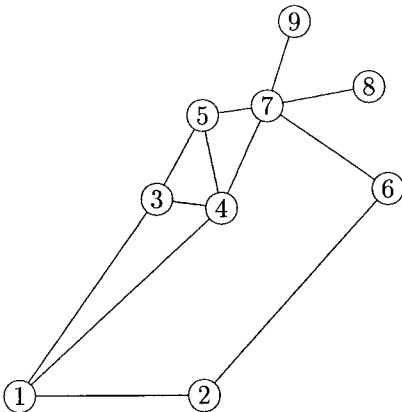


Figure 3.5: The SA MPLS network

The network carries 4 service classes and the service rates are $\mu_1 = 10$, $\mu_2 = \mu_3 = 20$ and $\mu_4 = 40$. The bi-directional packet arrival rates are given in table 3.2. The class-dependent packet arrival rates are $\lambda_{j1} = \omega_j$, $\lambda_{j2} = \lambda_{j3} = 2\omega_j$ and $\lambda_{j4} = 4\omega_j$ where ω_j is the arrival rate in table 3.2.

	2	3	4	5	6	7	8	9
1	2	1	1	0.5	1	3	0.5	0.5
2		0	0.5	0	1	0	0	0
3			2	1	0	0	0	0
4				0.5	0.5	1	0	0
5					0	0.5	0	0
6						1	0	0
7							1	1
8								0

Table 3.2: The packet arrival rates for the SA network

The optimal routing pattern sends packets along the LSPs in table 3.3 and yields an expected

delay of $T = 1.03$ time units. The algorithm requires two minutes of CPU time on a Pentium-II 350 MHz machine.

O-D pair	LSPs	s_{rj}
(1, 2)	1 – 2	1.00
(1, 3)	1 – 3	1.00
(1, 4)	1 – 4	1.00
(1, 5)	1 – 3 – 5	1.00
(1, 6)	1 – 2 – 6	1.00
(1, 7)	1 – 4 – 7	0.31
	1 – 3 – 5 – 7	0.40
	1 – 2 – 6 – 7	0.29
(1, 8)	1 – 4 – 7 – 8	0.31
	1 – 3 – 5 – 7 – 8	0.40
	1 – 2 – 6 – 7 – 8	0.29
(1, 9)	1 – 4 – 7 – 9	0.31
	1 – 3 – 5 – 7 – 9	0.40
	1 – 2 – 6 – 7 – 9	0.29
(2, 4)	2 – 1 – 4	0.01
	2 – 6 – 7 – 4	0.99
(2, 6)	2 – 6	1.00
(3, 4)	3 – 4	1.00
(3, 5)	3 – 5	1.00
(4, 5)	4 – 5	1.00
(4, 6)	4 – 7 – 6	1.00
(4, 7)	4 – 7	1.00
(5, 7)	5 – 7	1.00
(6, 7)	6 – 7	1.00
(7, 8)	7 – 8	1.00
(7, 9)	7 – 9	1.00

Table 3.3: The LSPs used by the SA network

3.6 Conclusion

This chapter showed how a variant of the flow deviation algorithm could be used to optimise the QoS offered by a specific type of multi-service MPLS network. The type of MPLS network considered here has such well-dimensioned links that all link delays can be ignored. However, the queueing systems in the LSRs have only limited capacities. Consequently, major delays can occur in the LSRs if the packets are not routed optimally. This chapter showed that a variant of the flow deviation algorithm could be used to efficiently “spread” the packet flows evenly throughout the network. This balances the loads offered to the LSRs, which in turns leads to an optimal network QoS.

Chapter 4

Minimising Link Delays in MPLS Networks

This chapter considers MPLS networks in which the links are no longer assumed to have “infinite” capacity. The mathematical model of such a network is the same as the general MPLS network model from chapter 2, except that the *cost function* is now specified. The cost function is the *expected packet delay*. Similar to the MPLS model in chapter 3, the network operator switches packet flows onto optimal LSPs in order to minimise the expected packet delay. However, in the current network the delays occur in the links and the *nodal delays* are assumed to be negligible.

Section 4.1 briefly states the mathematical model for this MPLS network. The expected packet delay is analysed in section 4.2. The *link delay* is quantified in section 4.3. Section 4.4 is concerned with the actual algorithms used to minimise the link delays. Finally, in section 4.5 the algorithms are analysed with reference to numerical results.

4.1 The Model

The label switched network has N nodes which are numbered from one and identified by their numbers. Let $\mathcal{N} = \{1, 2, \dots, N\}$ denote the set of nodes. Each node corresponds to a *label switching router* (LSR) which receives packets on incoming links and forwards them on appropriate outgoing links. The length of a packet¹ is an exponential random variable with mean $1/\mu$.

Each O-D pair is assigned a unique integer. Let \mathcal{J} be the set of all O-D pairs (identified by their numbers). Therefore $J = |\mathcal{J}| = N(N - 1)$. Since each physical link corresponds to an O-D pair, the links are identified by the numbers of the O-D pairs which they connect. If \mathcal{L} denotes the set of links (identified by their link numbers) then $\mathcal{L} \subseteq \mathcal{J}$. The function $\mathcal{L} : \mathcal{N} \times \mathcal{N} \mapsto \mathcal{J}$ is defined

¹Note that the packet length is not class-based as in chapter 3.

as $\mathcal{L}(o, d) = j$ if O-D pair (o, d) is numbered j .

Let C_ℓ denote the (physical) capacity (zero if the physical link does not exist) of the uni-directional link $i - j$ where $\mathcal{L}(i, j) = \ell$. The links have finite capacities and can therefore become overloaded. This results in link delays. The set \mathcal{L} can be expressed as (using the introduced notation)

$$\mathcal{L} = \{j \in \mathcal{J} \mid C_j > 0\}.$$

Let λ_j denote the arrival rate of packets to O-D pair j . The arrival process is assumed to be Poisson. Let λ be the total packet arrival rate to the network, therefore

$$\lambda = \sum_{j=1}^J \lambda_j.$$

λ is also referred to as the total *external traffic* attempting to enter the network.

A *label switched path* (LSP) $r = \langle o, d \rangle$ connecting nodes o and d is a sequence of physical links $o - o_1, o_1 - o_2, \dots, o_m - d$. The expression $\ell \in r$ indicates that link ℓ is traversed by LSP r . Note the notation distinction: (o, d) denotes the O-D pair, $o - d$ denotes the physical link and $\langle o, d \rangle$ denotes an LSP connecting nodes o and d . Let \mathcal{R}_j denote the set of LSPs (including the direct link if it exists) between O-D pair j . Let \mathcal{R} be the set of all routes in the network:

$$\mathcal{R} = \bigcup_{j \in \mathcal{J}} \mathcal{R}_j.$$

Let \mathcal{A}_ℓ denote the set of LSPs that traverse link ℓ for each $\ell \in \mathcal{L}$. Thus \mathcal{R}_j and \mathcal{A}_ℓ are defined as in chapter 2.

4.2 Analysis of the Link Delays

All packets offered to O-D pair j are switched along LSPs in \mathcal{R}_j . Let s_{rj} be the portion of the packets offered to O-D pair j which travel along LSP r . Note that $s_{rj} = 0$ if $r \notin \mathcal{R}_j$. If ν_r is the arrival rate of packets to LSP r then $\nu_r = s_{rj} \lambda_j$ for all $j \in \mathcal{J}$ and $r \in \mathcal{R}_j$. Since all packets offered to O-D pair j are switched along LSPs in \mathcal{R}_j , one obtains

$$\lambda_j = \sum_{r \in \mathcal{R}_j} \nu_r \quad (30)$$

$$\sum_{r \in \mathcal{R}_j} s_{rj} = 1 \quad (31)$$

for all $j \in \mathcal{J}$.

Let γ_ℓ denote the rate at which packets arrive to link ℓ . We assume that the γ_ℓ 's associated with different links are independent (*cf* the flow independence assumption of chapter 2). The relation between γ_ℓ and ν_r is

$$\gamma_\ell = \sum_{r \in \mathcal{A}_\ell} \nu_r \quad \text{for all } \ell \in \mathcal{L}. \quad (32)$$

Let the random variable D represent the delay a packet experiences in the network and let $T = \mathbf{E}[D]$. Let the random variable d_j represent the delay a packet offered to O-D pair j experiences and let $t_j = \mathbf{E}[d_j]$. It follows that

$$T = \sum_{j=1}^J \frac{\lambda_j}{\lambda} t_j \quad (33)$$

which is a decomposition of the expected packet delay in terms of O-D pair.

Define the random variable D_ℓ as the link delay which comprises the time that a packet spends waiting in the queue and using link ℓ . Since the link capacities C_ℓ are assumed to be the main sources of delay in the network, we only consider² link delays and the expected values $T_\ell = \mathbf{E}[D_\ell]$ and t_j are therefore related by

$$t_j = \sum_{r \in \mathcal{R}_j} s_{rj} \sum_{\ell \in r} T_\ell$$

and therefore (using (33)) the expected packet delay T can be expressed in terms of the link delays:

$$\begin{aligned} T &= \sum_{j=1}^J \frac{\lambda_j}{\lambda} t_j \\ &= \sum_{j=1}^J \frac{\lambda_j}{\lambda} \sum_{r \in \mathcal{R}_j} s_{rj} \sum_{\ell \in r} T_\ell \end{aligned}$$

²The propagation delays are assumed to be negligible.

$$\begin{aligned}
 &= \sum_{j=1}^J \frac{1}{\lambda} \sum_{r \in \mathcal{R}_j} s_{rj} \lambda_j \sum_{\ell \in r} T_\ell \\
 &= \sum_{j=1}^J \frac{1}{\lambda} \sum_{r \in \mathcal{R}_j} \nu_r \sum_{\ell \in r} T_\ell \\
 &= \sum_{r \in \mathcal{R}} \frac{1}{\lambda} \nu_r \sum_{\ell \in r} T_\ell \\
 &= \sum_{\ell=1}^L \frac{1}{\lambda} T_\ell \sum_{r \in \mathcal{A}_\ell} \nu_r \\
 &= \sum_{\ell=1}^L \frac{\gamma_\ell}{\lambda} T_\ell,
 \end{aligned} \tag{34}$$

where equation (32) has been used. This is a decomposition of the expected packet delay in terms of the expected link delays.

4.3 A Quantitative Measure of the Link Delay

The exact form of (34) depends on the form of T_ℓ . There are several criteria for selecting an appropriate link delay function. There are many functions which satisfy some of these criteria. The choice of link delay function is discussed in detail by Burns *et al.* [6]. We model the link as an $M/M/1$ queue with arrival rate γ_ℓ and C_ℓ servers each serving at rate μ . Thus the link delay T_ℓ is the expected waiting time

$$T_\ell = \frac{1}{\mu C_\ell - \gamma_\ell}. \tag{35}$$

4.4 Finding Optimal Label Switched Paths

This section considers optimal *LSP discovery* (finding optimal LSPs for all O-D pairs in an MPLS network). The optimal LSPs are expressed in terms of the set \mathcal{R} of LSPs and the factors s_{rj} which specify along which LSPs the packets are to be switched in order to minimise the expected packet delay. The flow deviation algorithms from chapter 2 can be used for LSP discovery (which can be stated as a non-linear optimisation problem). The decision variables depend on the flow deviation algorithm used. The optimisation problem solved by the Kleinrock algorithm uses the link flows γ_ℓ as the decision variables. Thus the object is to calculate an optimal *link flow vector* $\gamma = (\gamma_\ell)_{\ell \in \mathcal{L}}$ as explained in chapter 2, ie the following optimisation problem:

Optimisation Problem 4.1.

Minimise:

$$T(\gamma) = \sum_{\ell=1}^L \frac{\gamma_{\ell}}{\lambda} T_{\ell}$$

Subject to:

$$\begin{aligned} \lambda_j &= \sum_{r \in \mathcal{R}_j} \nu_r && \text{for all } j \in \mathcal{J} \\ \gamma_{\ell} &\geq 0 && \text{for all } \ell \in \mathcal{L}. \end{aligned}$$

The Bertsekas-Gallager algorithm solves the following optimisation problem.

Optimisation Problem 4.2.

Minimise:

$$\tilde{T}(\tilde{\nu})$$

Subject to:

$$\nu_r \geq 0 \quad \text{for all } j \in \mathcal{J}, r \in \mathcal{R}_j \text{ and } r \neq r_j$$

which is obtained as explained in chapter 2.

The Kleinrock and Bertsekas-Gallager algorithms both require the existence of a *feasible link flow vector*. This is a necessary and sufficient condition for the network to reach equilibrium and for (35) to be valid (see Cooper [12] or Wolff [34], for example).

Definition 4.1 (Feasible Link Flow Vector). A link flow vector γ is feasible if for each link $\ell \in \mathcal{L}$

$$\gamma_{\ell} < C_{\ell} \mu.$$

The flow deviation algorithms require the objective function T to be convex. Substituting (35) into (34) yields

$$\begin{aligned} T &= \sum_{\ell=1}^L \frac{\gamma_{\ell}}{\lambda} T_{\ell} \\ &= \sum_{\ell=1}^L \frac{\gamma_{\ell}}{\lambda (\mu C_{\ell} - \gamma_{\ell})} \end{aligned}$$

and therefore using the independence assumption

$$\begin{aligned}\frac{\partial T}{\partial \gamma_\ell} &= \frac{\mu C_\ell}{\lambda(\mu C_\ell - \gamma_\ell)^2} \\ \frac{\partial^2 T}{\partial (\gamma_\ell)^2} &= \frac{2\mu C_\ell}{\lambda(\mu C_\ell - \gamma_\ell)^3}\end{aligned}$$

for all $\ell \in \mathcal{L}$. These equations imply that the first two partial derivatives of T with respect to γ_ℓ are non-negative for all $\ell \in \mathcal{L}$. Therefore T is a convex function of γ_ℓ .

The result of the line search in the Kleinrock algorithm is used to construct a new link flow vector during the execution of the algorithm. The object of the line search is to minimise the expected packet delay, therefore an infeasible link flow vector is never constructed (see chapter 2) during the execution of the algorithm. No line search is performed in the Bertsekas-Gallager algorithm, therefore care must be taken not to construct an infeasible link flow vector γ . Kerschenbaum [21] prevents this by replacing

$$\omega_r = \eta \frac{c_r - c_{r_j}}{c_r^2}$$

in the iterative step

$$\nu_r^m = \max \{0, \nu_r^{m-1} - \omega_r\}$$

by

$$\omega_r = \min \left\{ \nu_r^{m-1}, \eta \frac{c_r - c_{r_j}}{c_r^2}, \zeta_{r_j} \right\}$$

where $r \in \mathcal{R}_j$ and the *minimum slack* on the least-cost LSP $r_j \in \mathcal{R}_j$ is

$$\zeta_{r_j} = \min \{C_\ell - \gamma_\ell \mid \ell \in r_j\}.$$

The complete iterative step (step 7 in algorithm 2.2) is now:

- Set $\phi^m \leftarrow \phi^{m-1}$ and perform the following procedure for each O-D pair j .
 - (a) Compute the cost rate vector \mathbf{c} , where $c_\ell = D'_\ell(\phi_\ell^m)$ for each $\ell \in \mathcal{L}$ and the second derivative cost rate vector \mathbf{c}^2 , where $c_\ell^2 = D''_\ell(\phi_\ell^m)$ for each $\ell \in \mathcal{L}$.

- (b) Let r_j be the least-cost LSP between O-D pair j . Then for each route $r \in \mathcal{R}_j$, $r \neq r_j$, set

$$\omega_r = \min \left\{ \nu_r^{m-1}, \eta \frac{c_r - c_{r_j}}{c_r^2}, \zeta_{r_j} \right\}$$

where

$$\zeta_{r_j} = \min \{ C_\ell - \gamma_\ell \mid \ell \in r_j \}.$$

and set

$$\nu_r^m = \nu_r^{m-1} - \omega_r.$$

- (c) Calculate the flow along the least-cost LSP connecting O-D pair j :

$$\nu_{r_j}^m = \lambda_j - \sum_{\substack{r \in \mathcal{R}_j \\ r \neq r_j}} \nu_r^m.$$

- (d) Update the link flow vector ϕ^m , where

$$\phi_\ell^m = \sum_{r \in \mathcal{A}_\ell} \nu_r^m \quad \text{for each } \ell \in \mathcal{L}.$$

Decrement the step-size η in an appropriate way.

The Kleinrock and Bertsekas-Gallager algorithms both start with feasible initial link flow vectors. The following algorithm (due to Kleinrock [23]) constructs an initial feasible link flow vector ϕ^0 .

Algorithm 4.1 (Constructing a feasible ϕ^0). Set $h_0 \leftarrow 1$ and initialise the cost rate vector \mathbf{c}

$$c_\ell = \frac{1}{\lambda \mu C_\ell} \quad \text{for each } \ell \in \mathcal{L}.$$

Use Dijkstra's algorithm to find the least-cost LSPs based on this cost rate vector. Let $\varphi^0 = (\varphi_\ell^0)_{\ell \in \mathcal{L}}$ denote the flow vector which results when, for each $j \in \mathcal{J}$, all flow λ_j is sent along the least-cost LSP connecting j .

Choose a feasible starting flow precision criterion ε (with $0 < \varepsilon < 1$), and two infeasibility criteria $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$.

1. $m \leftarrow 1$

2. Calculate β , where

$$\beta = \max \left\{ \frac{\varphi_\ell^{m-1}}{\mu} \mid \ell \in \mathcal{L} \right\}.$$

3. **if** $\beta < h_{m-1}$ **then**
 $\phi^0 \leftarrow \varphi^{m-1}/h_{m-1}$
 stop
 else
 continue
 endif

4. $h_m \leftarrow h_{m-1}(1 - \varepsilon(1 - \beta))/\beta$

5. Let ξ denote a flow vector that routes a total flow $h_m\lambda < \lambda$ through the network, where

$$\xi_\ell = \frac{h_m}{h_{m-1}} \varphi_\ell^{m-1} \quad \text{for all } \ell \in \mathcal{L}.$$

6. Compute the current cost rate vector \mathbf{c} , where

$$c_\ell = \left. \frac{\partial T}{\partial \gamma_\ell} \right|_{\gamma_\ell = \xi_\ell}$$

for each $\ell \in \mathcal{L}$.

7. Use Dijkstra's algorithm to find the least-cost LSPs based on the current cost rate vector \mathbf{c} . Denote the flow vector which results when, for each O-D pair j the flow $h_m\lambda_j$ is deviated along the least-cost LSP connecting j , by σ .
8. If the following conditions are both met no feasible flow vector can be found and the algorithm is terminated.

$$\left| \sum_{\ell=1}^L c_\ell (\sigma_\ell - \xi_\ell) \right| < \varepsilon_1$$

$$|h_m - h_{m-1}| < \varepsilon_2.$$

9. Find α (with $0 \leq \alpha \leq 1$) such that the flow vector $\gamma(\alpha) = (1 - \alpha)\xi + \alpha\sigma$ minimises T and set $\varphi^m = \gamma(\alpha)$.
10. $m \leftarrow m + 1$ and **return** to step 2.

This algorithm is motivated in chapter 3.

4.5 Results

This section presents some numerical results computed by the two flow deviation algorithms. The performance and the LSP configurations calculated by the Kleinrock and Bertsekas-Gallager algorithms are discussed.

4.5.1 The Convergence of the Flow Deviation Algorithms

The convergence of the two flow deviation algorithms is studied by considering their behaviour in two test networks (figures 4.1 and 4.2) by Villamizar [32].



Figure 4.1: The Topology of the 10 Node Network

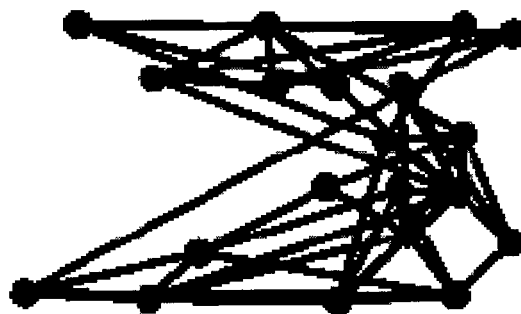


Figure 4.2: The Topology of the 20 Node Network

Figures 4.3 and 4.5 compare the convergence of the Kleinrock and Bertsekas-Gallager algorithms for these two networks. Iteration 0 represents the initial LSP configuration and T the expected packet delay. The Bertsekas-Gallager algorithm requires fewer iterations to converge than the Kleinrock algorithm in both test networks. The Kleinrock algorithm converges after 292 iterations and the Bertsekas-Gallager algorithm after 17 iterations for the 10 node network. The Kleinrock

algorithm converges after 740 iterations and the Bertsekas-Gallager algorithm after 18 iterations for the 20 node network. The behaviour of the Bertsekas-Gallager algorithm from iteration 1 until convergence is extracted from figures 4.3 and 4.5 and presented in figures 4.4 and 4.6. The step-size in the Bertsekas-Gallager algorithm was kept fixed at $\eta = 1$.

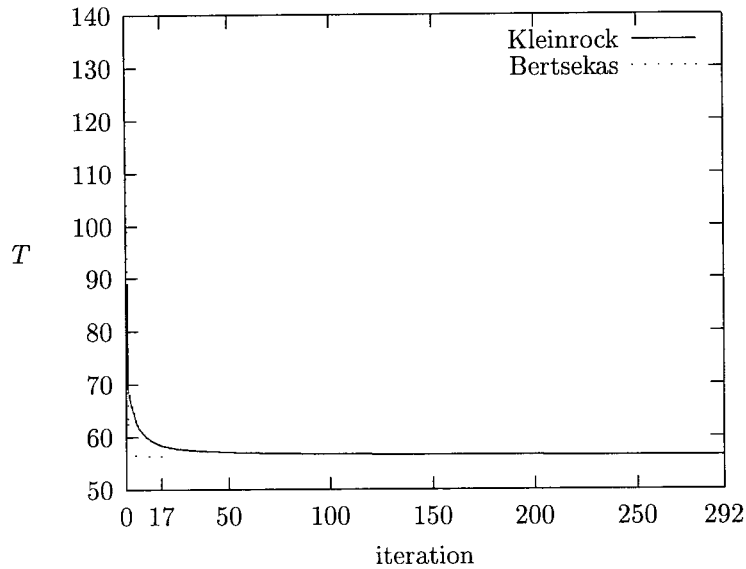


Figure 4.3: Convergence for the 10 Node Network

4.5.2 The LSP Sets \mathcal{R}

The LSP set \mathcal{R} is the main result of the flow deviation algorithm. The properties of the constructed LSP sets are now examined.

Let \mathcal{R}^k denote the set of LSPs discovered by the Kleinrock algorithm and \mathcal{R}^b denote the set of LSPs discovered by the Bertsekas-Gallager algorithm. We now consider the correlation between these LSP sets. A lucid graphical representation of this correlation is a pie chart. Both the LSPs discovered and the flows on these LSPs are compared. The significance of each pie chart slice is explained by referring to figure 4.7 which contains the pie charts for the 10 node network. Consider the LSP pie chart first. The slice marked " $\mathcal{R}^k \cap \mathcal{R}^b$ (strong)" represents the LSPs discovered by both algorithms and which are such that the flows ν_r on the corresponding LSPs differ by less than 5%. The slice marked " $\mathcal{R}^k \cap \mathcal{R}^b$ (weak)" represents the LSPs discovered by both algorithms and which are such that the flows ν_r on the corresponding LSPs differ by more than 5%. The slice marked " $\mathcal{R}^k \setminus \mathcal{R}^b$ " represents the LSPs discovered only by the Kleinrock algorithm. Finally, the slice marked " $\mathcal{R}^b \setminus \mathcal{R}^k$ " (which is not present in this particular example) in figure 4.8 represents the LSPs discovered only by the Bertsekas-Gallager algorithm. Consider the flow pie chart. The slice marked " $\mathcal{R}^k \cap \mathcal{R}^b$ (strong)" represents the flows on LSPs discovered by both algorithms and which are such that the flows ν_r on the corresponding LSPs differ by less than 5%. The slice

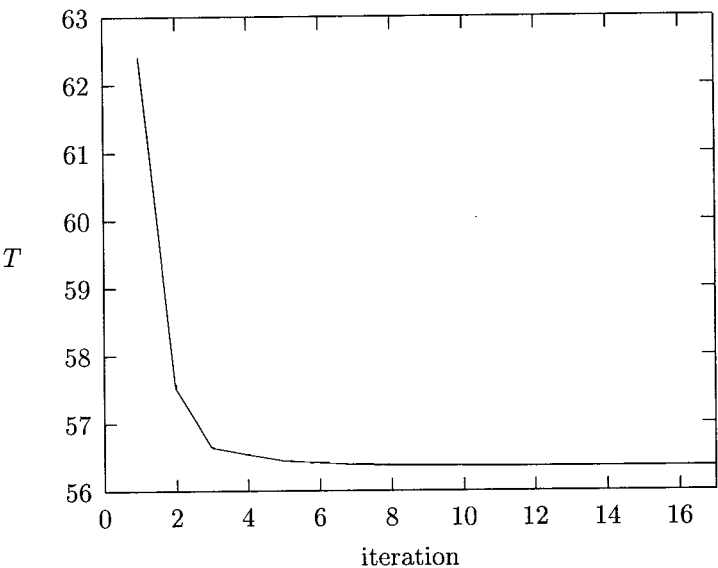


Figure 4.4: Convergence of the Bertsekas-Gallager Algorithm for the 10 Node Network

marked “ $\mathcal{R}^k \cap \mathcal{R}^b$ (weak)” represents the flows ν_r on LSPs discovered by both algorithms and which are such that the flows ν_r on the corresponding LSPs differ by more than 5%. Let r be an LSP discovered by both algorithms and ν_r^k be the flow which the Kleinrock algorithm sends along r and let ν_r^b be the flow which the Bertsekas-Gallager algorithm sends along r . The flow contribution of r in the pie chart is then given by $\max \{ \nu_r^k, \nu_r^b \}$. The slice marked “ $\mathcal{R}^k \setminus \mathcal{R}^b$ ” represents the flow on LSPs only discovered by the Kleinrock algorithm. The percentages represented by the slices in all the pie charts are given in table 4.1.

	$\mathcal{R}^k \cap \mathcal{R}^b$ (strong)		$\mathcal{R}^b \cap \mathcal{R}^k$ (weak)		$\mathcal{R}^k \setminus \mathcal{R}^b$		$\mathcal{R}^b \setminus \mathcal{R}^k$	
network	LSPs	flow	LSPs	flow	LSPs	flow	LSPs	flow
10 node	46%	80%	32%	19%	22%	1%	0%	0%
20 node	56%	74.25%	31.5%	25%	12%	0.75%	0.5%	0%
50 node	69%	85%	19%	14%	12%	1%	0%	0%
100 node	47%	71%	24%	24%	29%	5%	0%	0%

Table 4.1: LSP Correlation

It is evident from figures 4.7 and 4.8 that although not all LSPs are discovered by both algorithms, the largest portion of the total flow λ is sent along LSPs discovered by both. Note in figure 4.8, for example, that although 0.5% of the LSPs are only discovered only by the Bertsekas-Gallager algorithm, a negligible portion of the total flow λ is sent along these LSPs (too small a portion even to be displayed in the pie chart).

We also compare the performance of the algorithms in more detail for a 50 node network by Villamizar [32] (figure 4.9) and a 100 node network (figure 4.10) generated by the method presented

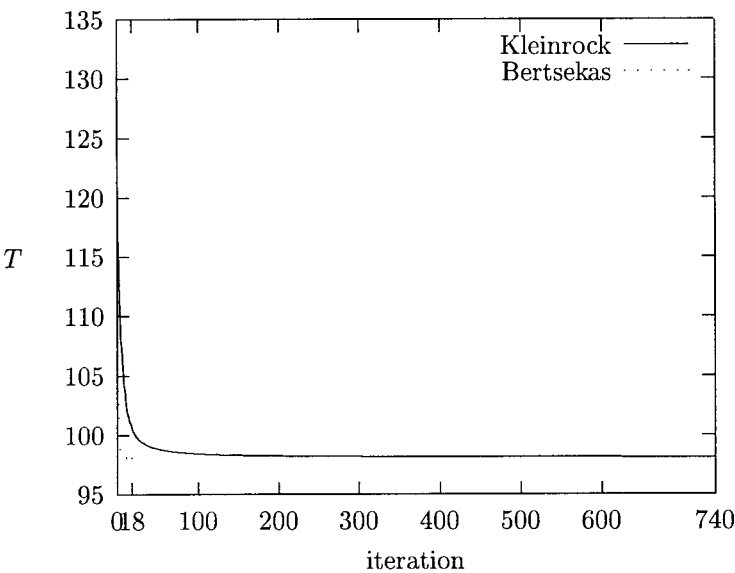


Figure 4.5: Convergence for the 20 Node Network

by Arvidsson [4]. The 100 node network is planar.

The pie charts for these networks are shown in figures 4.11 and 4.12. 12% and 29% respectively, of the LSPs discovered in these networks are only discovered by the Kleinrock algorithm. However, only 1% and 5% respectively, of the total flow λ is sent along these LSPs.

The LSP statistics for the Kleinrock and Bertsekas-Gallager algorithms for the 50 node network are in tables 4.2 and 4.3. The columns in the table are (for the k th row, from left to right) the length k of the LSP, the number of LSPs of length k , the percentage of LSPs of length k , the total flow on LSPs of length k , the average flow on an LSP of length k and the percentage of the total flow on LSPs of length k . It is interesting to note that both algorithms send the same portion of the total flow λ along LSPs of length k . However, the number of LSPs discovered by the two algorithms differ and therefore the average flow sent along LSPs of length k by the two algorithms differ. The last row in each table contains the length, average flow and percentage of the total flow carried by the average LSP.

The normalised LSP statistics are in tables 4.4 and 4.5. The *normalised length* of an LSP connecting O-D pair j is defined as the difference between the length of the LSP and the length of the least-cost LSP connecting j . The columns have the same meanings as in the case of the unnormalised statistics. Both algorithms send 97.45% of the total flow λ along LSPs of normalised length 0.

Note that after the completion of both algorithms, LSPs with ν_r smaller than 0.005 times the average LSP flow were discarded. The Kleinrock algorithm discovered 5213 LSPs and discarded 2068 LSPs to leave 3145 “active” LSPs. The Bertsekas-Gallager algorithm discovered 5195 LSPs

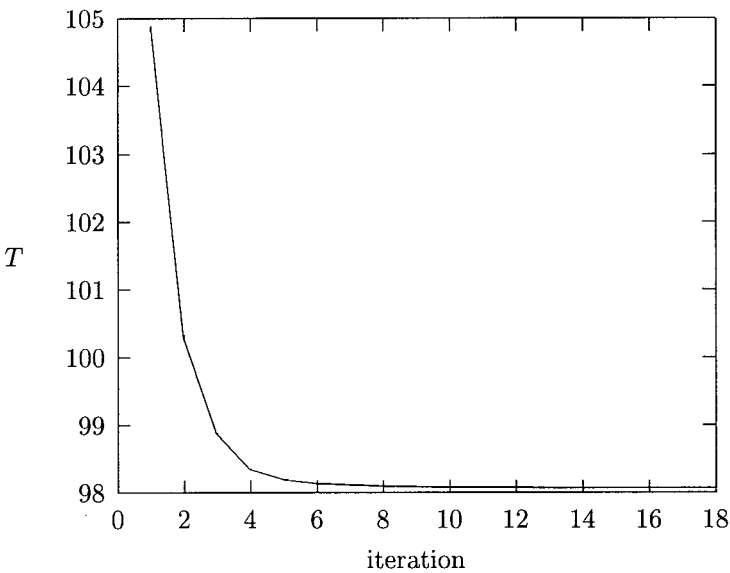


Figure 4.6: Convergence of the Bertsekas-Gallager Algorithm for the 20 Node Network

and discarded 2420 LSPs to leave 2775 “active” LSPs. The Kleinrock algorithm yielded an expected packet delay of $T = 2.0225 \times 10^2$ and the Bertsekas-Gallager algorithm yielded an expected packet delay $T = 2.0224 \times 10^2$. All these figures are for the 50 node network.

length	count	%	flow	avg flow	%
1	201	6.39	5.76×10^5	2.87×10^3	8.74
2	593	18.85	1.55×10^6	2.61×10^3	23.53
3	1162	36.95	2.56×10^6	2.20×10^3	38.86
4	994	31.61	1.69×10^6	1.70×10^3	25.67
5	195	6.20	2.11×10^5	1.08×10^3	3.20
3.12			2.09×10^3		0.03

Table 4.2: LSP Statistics of the Kleinrock Algorithm for the 50 Node Network

The LSP multiplicity of the Kleinrock and the Bertsekas-Gallager algorithms for the 50 node network are in tables 4.6 and 4.7. The *LSP multiplicity* of O-D pair j is defined as the number of LSPs $|\mathcal{R}_j|$ connecting O-D pair j . The Kleinrock algorithm, for example, connects 1895 O-D pairs by one LSP only, whereas 447 O-D pairs are connected by 2 LSPs. The last line of each table contains the average number of LSPs connecting an O-D pair.

The LSP and normalised LSP statistics for the 100 node network are in tables 4.8 to 4.11. The Kleinrock algorithm discovered 14342 LSPs and discarded 18 LSPs to leave 14324 “active” LSPs. The Bertsekas-Gallager algorithm discovered 14134 LSPs and discarded 3928 LSPs to leave 10206 “active” LSPs. The Bertsekas-Gallager algorithm uses fewer “active” LSPs than the Kleinrock algorithm for both the 50 and 100 node networks. The Bertsekas-Gallager also sends more (45.46%)

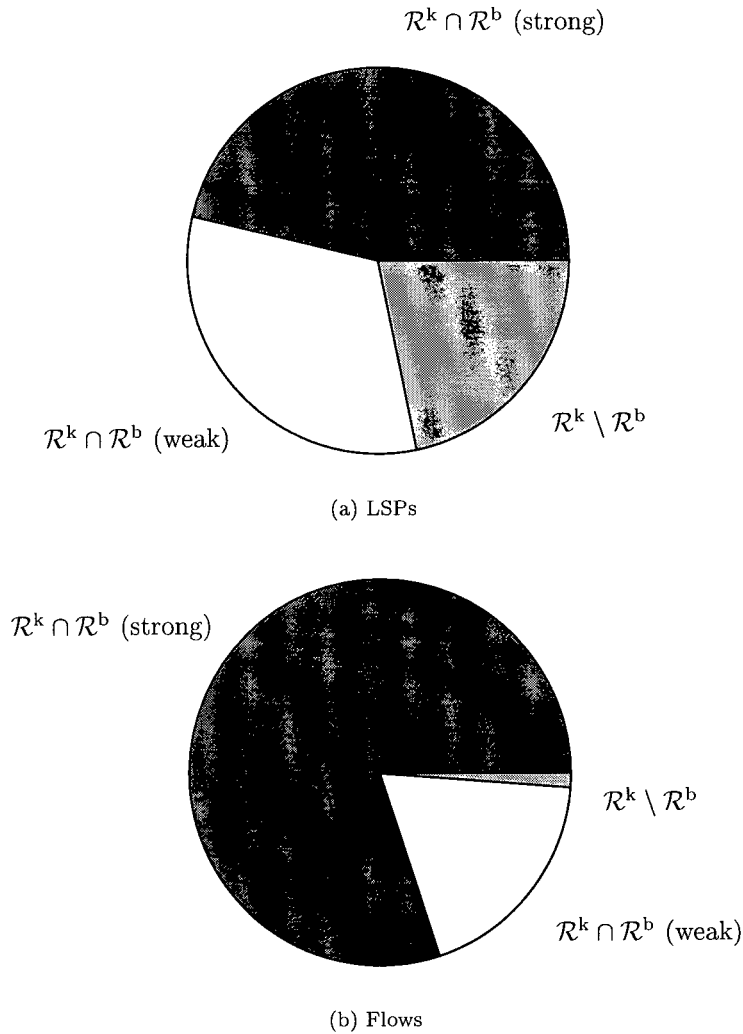


Figure 4.7: LSP Correlation for the 10 Node Network

of the total flow λ along LSPs of normalised length 0 than the Kleinrock algorithm (35.32%). The use of longer LSPs (by both algorithms) than in the 50 node network is due to the planarity of the 100 node network. Thus there are relatively fewer possible “short” LSPs to use than in the densely-meshed 50 node network. Figure 4.13 is a histogram of the percentage of least-cost LSPs (LSPs of normalised length 0) used by each algorithm in each of the four networks. The low usage of least-cost LSPs in the planar 100 node network as opposed to the other densely-meshed networks is evident.

The LSP multiplicity for the Kleinrock and the Bertsekas-Gallager algorithms are in tables 4.12 and 4.13. The Bertsekas-Gallager algorithm connects the average O-D pair by 1.03 LSPs as opposed to the 1.45 LSPs employed by the Kleinrock algorithm. This implies that the Kleinrock algorithm provides each O-D pair with more possible backup LSPs (which can be used in case of an LSP

length	count	%	flow	avg flow	%
1	201	7.24	5.76×10^5	2.87×10^3	8.74
2	580	20.90	1.55×10^6	2.67×10^3	23.53
3	1049	37.80	2.56×10^6	2.44×10^3	38.86
4	806	29.05	1.69×10^6	2.09×10^3	25.67
5	139	5.01	2.11×10^5	1.51×10^3	3.20
3.04			2.37×10^3		0.04

Table 4.3: LSP Statistics of the Bertsekas-Gallager Algorithm for the 50 Node Network

length	count	%	flow	avg flow	%
0	3002	95.45	6.41×10^6	2.14×10^3	97.45
1	143	4.55	1.68×10^5	1.18×10^3	2.55
0.05			2.09×10^3		0.03

Table 4.4: Normalised LSP Statistics of the Kleinrock Algorithm for the 50 Node Network

failure due to one or more of its links being down for some reason) than the Bertsekas-Gallager algorithm.

length	count	%	flow	avg flow	%
0	2681	96.61	6.41×10^6	2.39×10^3	97.45
1	94	3.39	1.68×10^5	1.79×10^3	2.55
0.03			2.37×10^3		0.04

Table 4.5: Normalised LSP Statistics of the Bertsekas-Gallager Algorithm for the 50 Node Network

number of LSPs	number of O-D pairs
1	1895
2	447
3	80
4	26
5	1
6	0
7	1
1.28	

Table 4.6: LSP Multiplicity of the Kleinrock Algorithm for the 50 Node Network

number of LSPs	number of O-D pairs
1	2189
2	209
3	42
4	8
5	2
1.13	

Table 4.7: LSP Multiplicity of the Bertsekas-Gallager Algorithm for the 50 Node Network

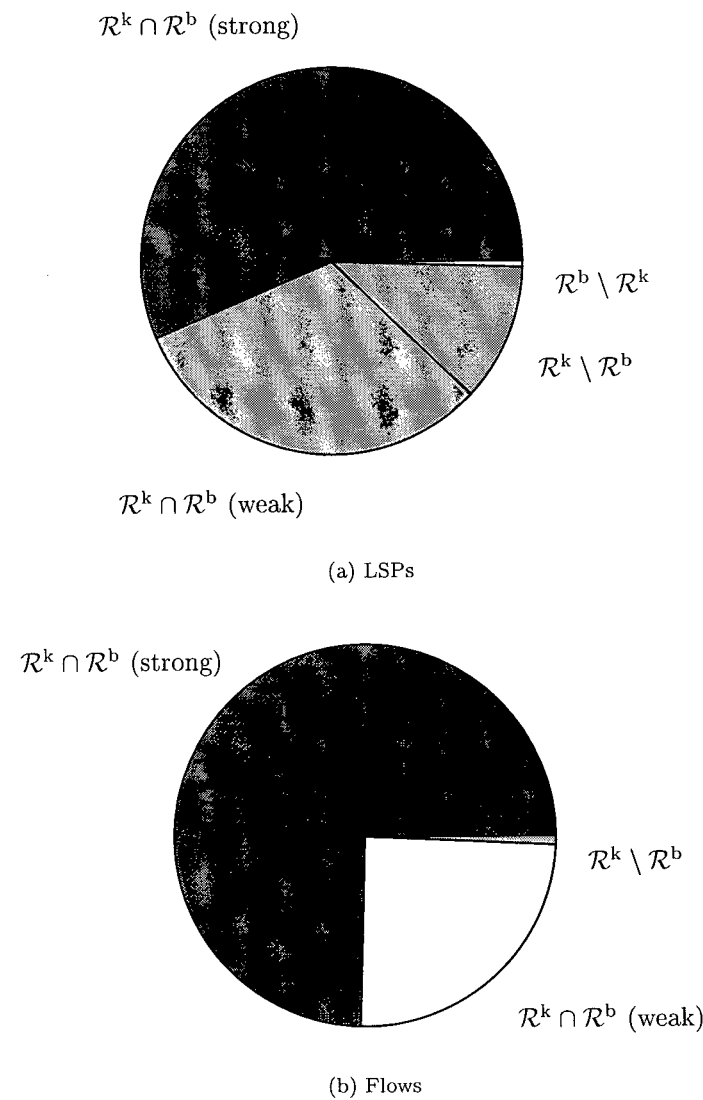


Figure 4.8: LSP Correlation for the 20 Node Network

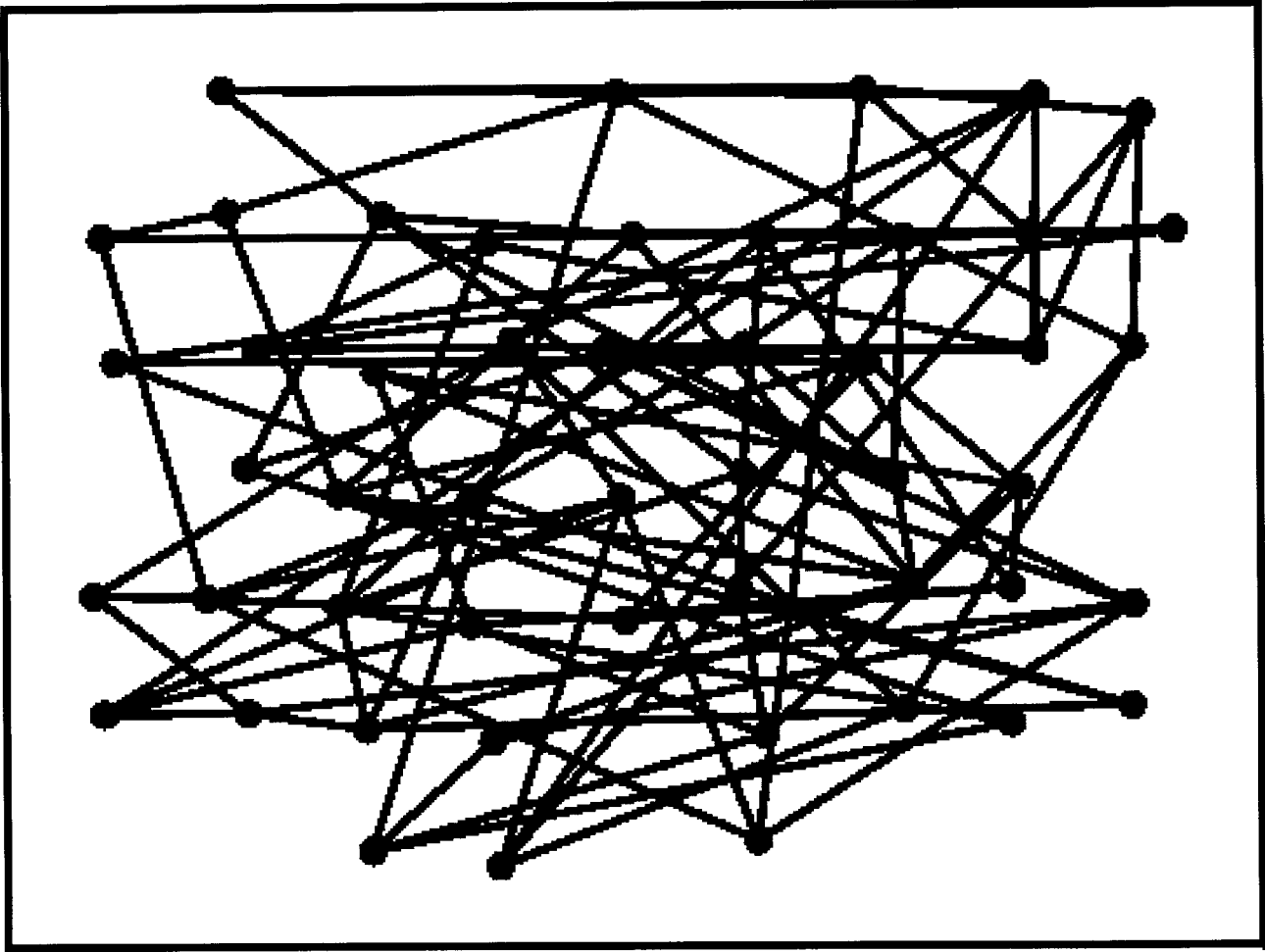


Figure 4.9: The Topology of the 50 Node Network

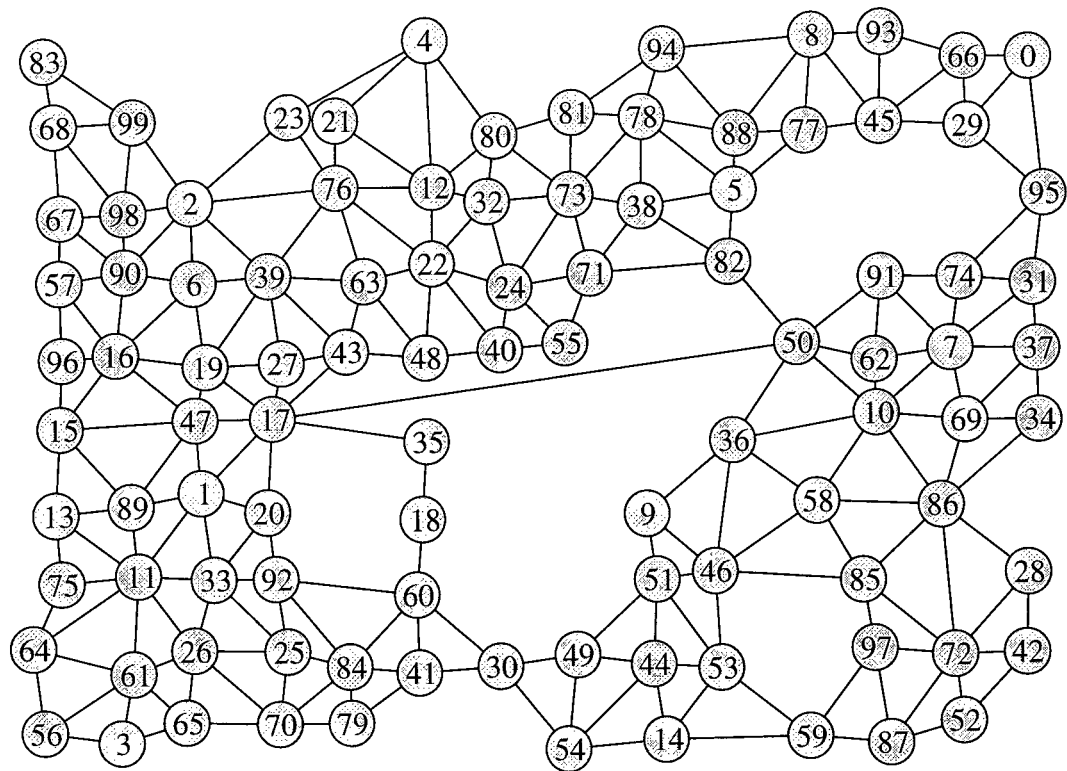


Figure 4.10: The Topology of the 100 Node Network

length	count	%	flow	avg flow	%
1	336	2.35	3.66×10^3	1.09×10^1	14.65
2	642	4.48	3.89×10^3	6.07	15.57
3	958	6.69	3.67×10^3	3.84	14.69
4	1288	8.99	3.22×10^3	2.50	12.89
5	1600	11.17	2.78×10^3	1.74	11.12
6	1816	12.68	2.40×10^3	1.32	9.60
7	1852	12.93	1.88×10^3	1.02	7.52
8	1694	11.83	1.40×10^3	8.26×10^{-1}	5.60
9	1454	10.15	9.54×10^2	6.56×10^{-1}	3.82
10	1110	7.75	5.56×10^2	5.01×10^{-1}	2.22
11	724	5.05	3.05×10^2	4.21×10^{-1}	1.22
12	452	3.15	1.64×10^2	3.62×10^{-1}	0.66
13	226	1.58	6.89×10^1	3.05×10^{-1}	0.28
14	106	0.74	2.94×10^1	2.77×10^{-1}	0.12
15	48	0.33	8.87	1.85×10^{-1}	0.04
16	14	0.10	1.14	8.13×10^{-2}	0.00
17	4	0.03	1.33×10^{-1}	3.34×10^{-2}	0.00
6.85			1.75		7.00×10^{-5}

Table 4.8: LSP Statistics of the Kleinrock Algorithm for the 100 Node Network

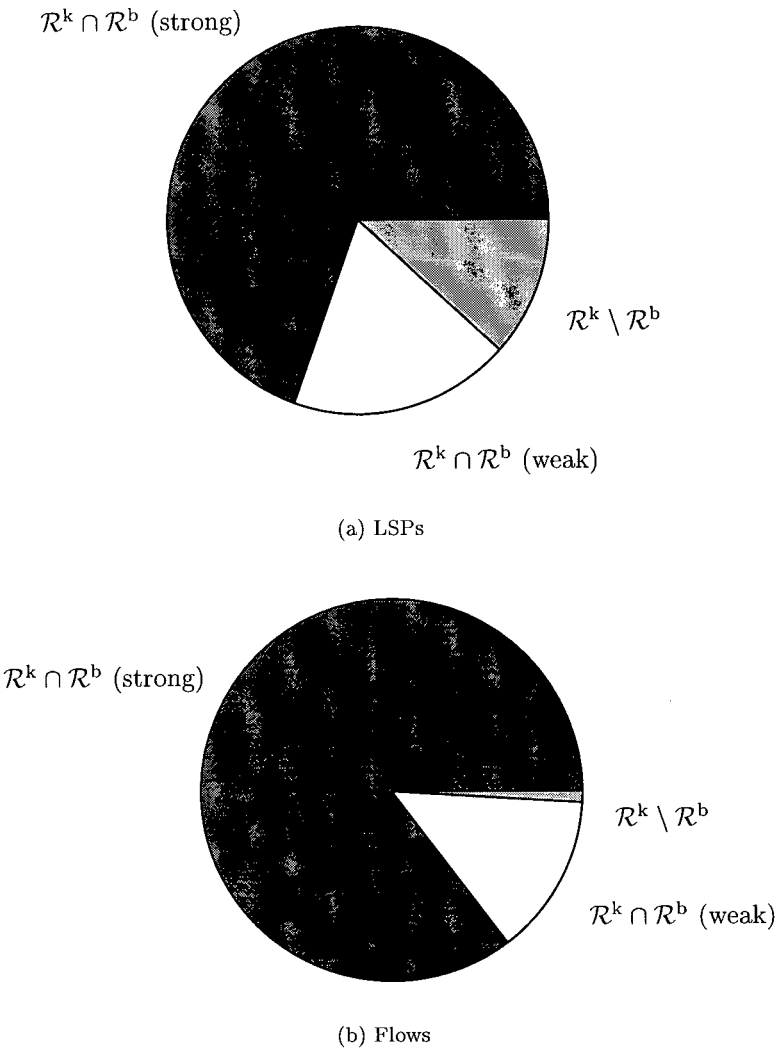


Figure 4.11: LSP Correlation for the 50 Node Network

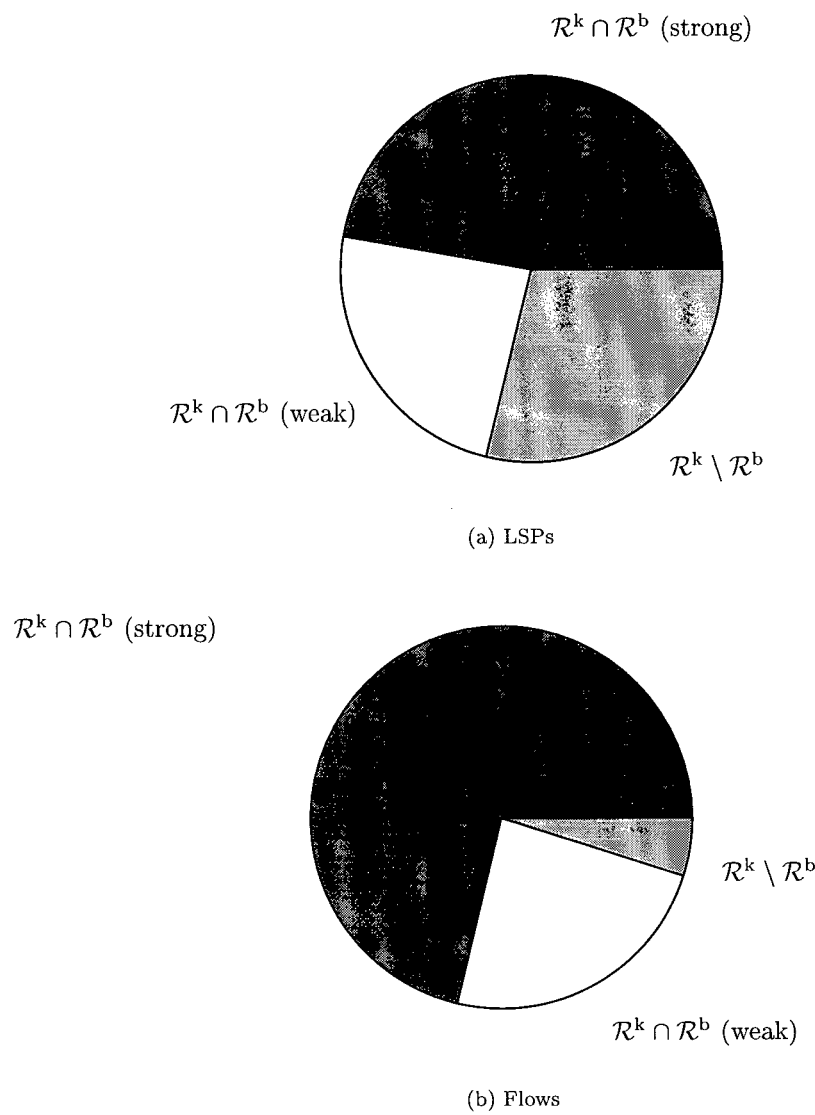


Figure 4.12: LSP Correlation for the 100 Node Network

length	count	%	flow	avg flow	%
1	331	3.24	3.66×10^3	1.11×10^1	14.64
2	604	5.92	3.90×10^3	6.46	15.60
3	872	8.54	3.67×10^3	4.21	14.68
4	1086	10.64	3.22×10^3	2.97	12.88
5	1268	12.43	2.77×10^3	2.18	11.08
6	1372	13.44	2.42×10^3	1.76	9.68
7	1283	12.57	1.88×10^3	1.46	7.52
8	1109	10.87	1.40×10^3	1.27	5.60
9	868	8.50	9.50×10^2	1.09	3.80
10	610	5.98	5.45×10^2	8.94×10^{-1}	2.18
11	386	3.78	3.16×10^2	8.17×10^{-1}	1.26
12	237	2.32	1.62×10^2	6.85×10^{-1}	0.65
13	107	1.05	6.63×10^1	6.20×10^{-1}	0.26
14	49	0.48	2.93×10^1	5.98×10^{-1}	0.12
15	22	0.22	9.02	4.10×10^{-1}	0.04
16	2	0.02	5.00×10^{-1}	2.50×10^{-1}	0.01
6.31			2.45		9.80×10^{-5}

Table 4.9: LSP Statistics of the Bertsekas-Gallager Algorithm for the 100 Node Network

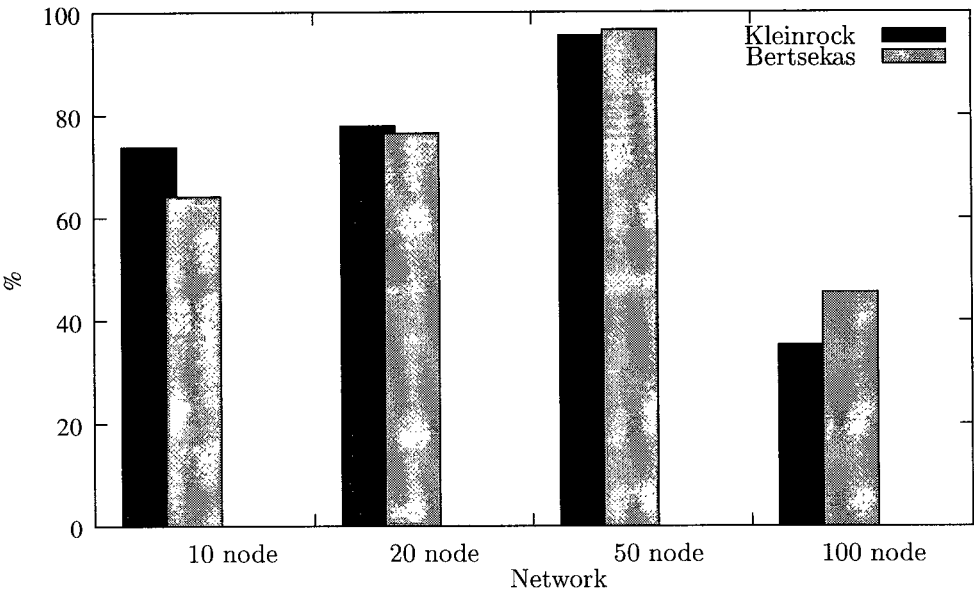


Figure 4.13: Percentage use of Shortest Possible LSPs

length	count	%	flow	avg flow	%
0	5060	35.32	1.38×10^4	2.74	55.30
1	3634	25.37	6.13×10^3	1.69	24.56
2	2424	16.92	3.00×10^3	1.24	12.02
3	1632	11.39	1.20×10^3	7.38×10^{-1}	4.81
4	974	6.80	5.55×10^2	5.70×10^{-1}	2.22
5	378	2.64	1.81×10^2	4.79×10^{-1}	0.73
6	156	1.10	7.32×10^1	4.69×10^{-1}	0.29
7	48	0.33	1.47×10^1	3.06×10^{-1}	0.06
8	14	0.10	1.47	1.05×10^{-1}	0.01
9	4	0.03	2.55×10^{-1}	6.38×10^{-2}	0.00
1.44			1.75		7.00×10^{-5}

Table 4.10: Normalised LSP Statistics of the Kleinrock Algorithm for the 100 Node Network

length	count	%	flow	avg flow	%
0	4640	45.46	1.39×10^4	3.00	55.58
1	2422	23.73	6.04×10^3	2.49	24.15
2	1580	15.48	3.07×10^3	1.94	12.28
3	925	9.06	1.17×10^3	1.26	4.68
4	419	4.10	5.53×10^2	1.32	2.21
5	154	1.51	1.94×10^2	1.26	0.78
6	56	0.55	7.10×10^1	1.27	0.28
7	9	0.10	7.03	7.81×10^{-1}	0.03
8	1	0.01	1.76	1.76	0.01
1.10			2.45		9.80×10^{-5}

Table 4.11: Normalised LSP Statistics of the Bertsekas-Gallager Algorithm for the 100 Node Network

number of LSPs	number of O-D pairs
1	6430
2	2794
3	478
4	124
5	68
6	6
1.45	

Table 4.12: LSP Multiplicity of the Kleinrock Algorithm for the 100 Node Network

number of LSPs	number of O-D pairs
1	9617
2	260
3	23
1.03	

Table 4.13: LSP Multiplicity of the Bertsekas-Gallager Algorithm for the 100 Node Network

4.6 Conclusion

This chapter applied a variant of the flow deviation algorithm to a model of an MPLS network in order to optimise the QoS. The network model is considered to be more realistic than the model of chapter 3 since the link delays are no longer regarded as “infinite”. Major delays can now occur in the links if the packets are not routed optimally. This chapter showed that the flow deviation algorithm provides the network operator with an efficient algorithm for “spreading” the packet flows evenly throughout the network and thus optimising the QoS offered by the network.

Chapter 5

Conclusion

This thesis considered QoS optimisation in MPLS networks. The expected packet delay was used as the QoS measure. Efficient algorithms were developed to optimally divide the network traffic into forwarding equivalence classes (FECs), to find optimal LSPs which minimise the expected packet delay and to switch these FECs along the optimal LSPs. These algorithms were applied to several test networks.

This thesis showed that efficient algorithms exist for computing LSP sets which optimise the network's QoS. The algorithms provided in this thesis perform this computation in near-real time and are therefore usable in real-world scenarios.

Appendix A

Convex and Concave Functions

Convex and *concave functions* play a major role in flow deviation. *Convex sets* are used in the characterisation of convex functions and are defined as follows (see Webb [33]).

Definition A.1 (Convex Set). A set S in \mathbb{R}^n is said to be *convex* if, whenever \mathbf{x} and \mathbf{y} are two points of S , the line segment $\{(1-t)\mathbf{x} + t\mathbf{y} \mid 0 \leq t \leq 1\}$ also lies in S .

The following two definitions are taken from Fischer [16].

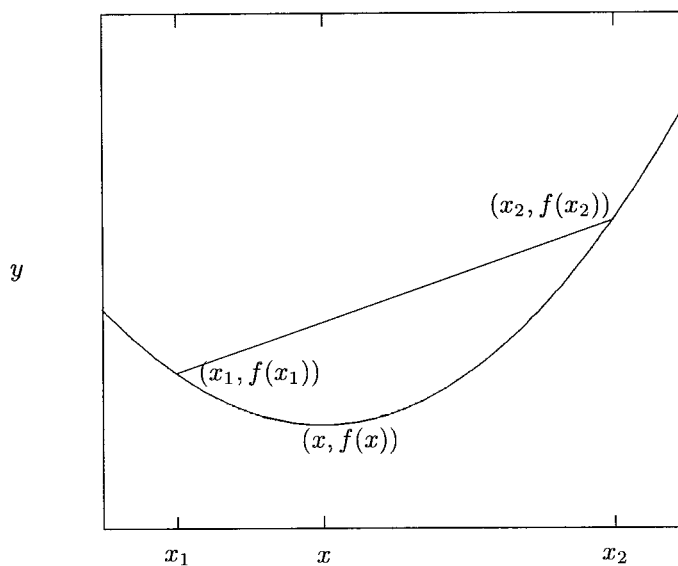


Figure A.1: A Convex Function

Definition A.2 (Convex Function). A real-valued function of a real variable which is defined on an interval I is called *convex* on I if and only if for x_1 and x_2 in I such that $x_1 < x_2$, we have

$$f(x) \leq f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1) \quad \text{for all } x \in [x_1, x_2].$$

Figure A.1 illustrates this concept.

Definition A.3 (Concave Function). Let I be an interval. We call f concave on I if and only if $x_1 \in I$, $x_2 \in I$ and $x_1 < x < x_2$ imply

$$f(x) \geq f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1).$$

Figure A.2 illustrates this concept.

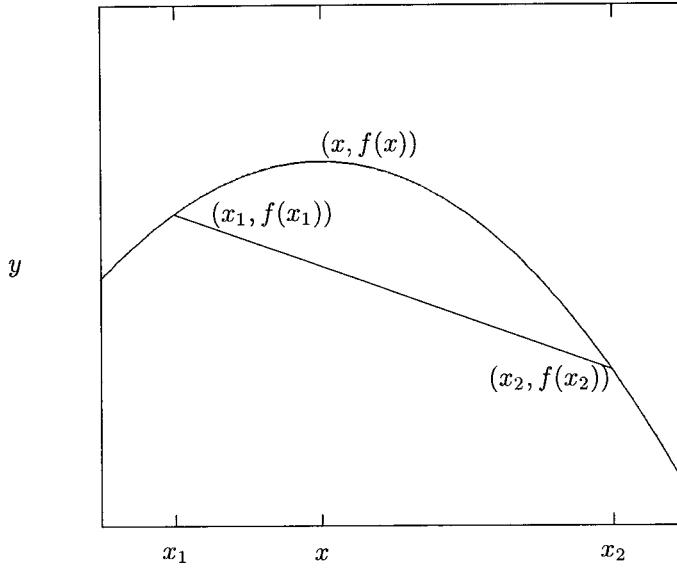


Figure A.2: A Concave Function

A characterisation of a convex function which is frequently used is that it satisfies Jensen's inequality (see Saaty *et al.* [29] for example). Let f be a real-valued function defined on the interval $I \subseteq \mathbb{R}$. Jensen's inequality states that

$$f\left(\frac{x_1 + x_2}{2}\right) \leq \frac{f(x_1) + f(x_2)}{2}$$

where $x_1, x_2 \in I$ and $x_1 < x_2$. Note that this is just a special case of Definition A.2 with $x = (x_1 + x_2)/2$. A more general version of Jensen's inequality [29] is

$$f((1 - \theta)x_1 + \theta x_2) \leq (1 - \theta)f(x_1) + \theta f(x_2)$$

where $\theta \in [0, 1]$. This is again a special case of Definition A.2 with $x = (1 - \theta)x_1 + \theta x_2$.

These definitions and characterisations are for functions of one variable. Convex and concave functions of several variables are generalisations of the above definitions. Let f be a real-valued function and S a convex subset of the domain of f . The following two definitions are due to Fleming [17].

Definition A.4 (Convex Function of Several Variables). *The function f is convex on S if, for every $\mathbf{x}_1, \mathbf{x}_2 \in S$ and $t \in [0, 1]$,*

$$f(t\mathbf{x}_1 + (1 - t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1 - t)f(\mathbf{x}_2).$$

Definition A.5 (Concave Function of Several Variables). *The function f is concave on S if, for every $\mathbf{x}_1, \mathbf{x}_2 \in S$ and $t \in [0, 1]$,*

$$f(t\mathbf{x}_1 + (1 - t)\mathbf{x}_2) \geq tf(\mathbf{x}_1) + (1 - t)f(\mathbf{x}_2).$$

Note that Definition A.4 is a generalisation of the second formulation of Jensen's inequality with $\theta = 1 - t$.

Fleming [17] provides a test for convexity which is stated in the following theorem.

Theorem A.1. *Let f be differentiable on a convex set S . f is convex on S if and only if*

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + df(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)$$

for every $\mathbf{x}_0, \mathbf{x} \in S$.

Proof. See Fleming [17]. □

An important property of a convex function is that any local minimum is a global minimum. Similarly, any local maximum of a concave function is a global maximum of the function. This property is frequently exploited in optimisation theory.

Theorem A.2. *Let f be differentiable and convex on an open convex set S with a local minimum at $\mathbf{x}_0 \in S$. f has an absolute minimum at \mathbf{x}_0 .*

Proof. f is differentiable at the point \mathbf{x}_0 (where it has a local minimum), therefore \mathbf{x}_0 is a critical point (see Fleming [17] for a proof of this property) and $df(\mathbf{x}_0) = \mathbf{0}$. Let \mathbf{x} be any point in S . By Theorem A.1

$$\begin{aligned} f(\mathbf{x}) &\geq f(\mathbf{x}_0) + df(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) \\ &= f(\mathbf{x}_0) + \mathbf{0} \cdot (\mathbf{x} - \mathbf{x}_0) \\ &= f(\mathbf{x}_0). \end{aligned}$$

Thus $f(\mathbf{x}) \geq f(\mathbf{x}_0)$ for every $\mathbf{x} \in S$ and consequently f has a global minimum at \mathbf{x}_0 . □

Appendix B

The Frank-Wolfe Method

This appendix discusses the *Frank-Wolfe method* on which flow deviation is based. Sections B.1 and B.2 list and develop some useful results from linear algebra and optimisation theory. Section B.3 discusses the actual Frank-Wolfe method.

B.1 Some Results from Linear Algebra

Optimisation theory relies heavily on linear algebra, particularly matrix theory. Some important concepts from matrix theory are reviewed in this section.

First of all we state some conventions which apply throughout this appendix. All vectors used in this work are real vectors with matrix representations either as column vectors or row vectors. The column vector representation is used for all vectors. Thus, if a vector is defined as $\mathbf{x} = (x_1, x_2, \dots, x_n)$, the matrix representation of \mathbf{x} is the column vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

When \mathbf{x} has to be used as a row vector, the transpose

$$\mathbf{x}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$$

is used.

The $n \times n$ identity matrix is denoted by \mathbf{I}_n and the $n \times n$ null matrix is denoted by $\mathbf{0}_n$. The null vector is denoted by $\mathbf{0}$ regardless of the dimension or whether a column or row vector is under consideration. The dimension is always evident from the context.

Definition B.1 (Transpose). Let A be an $m \times n$ matrix. The transpose of A , denoted by A^T , is the $n \times m$ matrix with the property that $[A^T]_{ij} = [A]_{ji}$. Thus A^T is the matrix which results when the rows and columns of A are interchanged.

The next theorem lists important properties of the transpose operation.

Theorem B.1. Let A and B be $m \times n$ matrices and C an $n \times \ell$ matrix. Then

1. $(A^T)^T = A$
2. $(A + B)^T = A^T + B^T$ and $(A - B)^T = A^T - B^T$
3. $(kA)^T = kA^T$ where $k \in \mathbb{R}$
4. $(AC)^T = C^T A^T$.

Proof. See Anton [3] for example. □

Definition B.2 (Quadratic Form). Let A be an $n \times n$ symmetric matrix and \mathbf{x} a column vector (an $n \times 1$ matrix). The quantity $\mathbf{x}^T A \mathbf{x}$ is called a quadratic form.

Let the $n \times n$ matrix A and the column vector \mathbf{x} be defined as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

respectively, then

$$\mathbf{x}^T A \mathbf{x} = \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j. \quad (36)$$

The partial derivatives of quadratic forms are used in the Frank-Wolfe method.

Lemma B.1. *Let A be an $n \times n$ symmetric matrix. The partial derivative of the quadratic form $\mathbf{x}^T A \mathbf{x}$ with respect to x_k is*

$$\frac{\partial}{\partial x_k} (\mathbf{x}^T A \mathbf{x}) = 2 \sum_{i=1}^n a_{ki} x_i.$$

Proof. (36) can be written as

$$\begin{aligned} \mathbf{x}^T A \mathbf{x} &= \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j \\ &= \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_i x_j + \sum_{i=1}^n a_{ii} x_i^2. \end{aligned}$$

Differentiation with respect to x_k yields

$$\begin{aligned} \frac{\partial}{\partial x_k} (\mathbf{x}^T A \mathbf{x}) &= \sum_{\substack{j=1 \\ j \neq k}}^n a_{kj} x_j + \sum_{\substack{i=1 \\ i \neq k}}^n a_{ik} x_i + 2a_{kk} x_k \\ &= \sum_{j=1}^n a_{kj} x_j + \sum_{i=1}^n a_{ik} x_i \\ &= 2 \sum_{i=1}^n a_{ki} x_i, \end{aligned}$$

where the symmetry of A has been used. □

The Frank-Wolfe method also makes use of the *gradient vector* of a function f of n variables, defined as (see Fleming [17] for example)

$$\nabla(f) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right).$$

Theorem B.2. *Let A be an $n \times n$ symmetric matrix. The gradient vector of the quadratic form $\mathbf{x}^T A \mathbf{x}$ is*

$$\nabla (\mathbf{x}^T A \mathbf{x}) = 2A\mathbf{x}.$$

Proof. This result follows directly from Lemma B.1. \square

We conclude this section by giving definitions of *positive definite* and *positive semi-definite* quadratic forms and matrices.

Definition B.3 (Positive Definite). Let A be an $n \times n$ symmetric matrix. The quadratic form $\mathbf{x}^T A \mathbf{x}$ is called *positive definite* if $\mathbf{x}^T A \mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$. The matrix A is called *positive definite* if the quadratic form $\mathbf{x}^T A \mathbf{x}$ is positive definite.

Definition B.4 (Positive Semi-Definite). Let A be an $n \times n$ symmetric matrix. The quadratic form $\mathbf{x}^T A \mathbf{x}$ is called *positive semi-definite* if $\mathbf{x}^T A \mathbf{x} \geq 0$ for all \mathbf{x} . The matrix A is called *positive semi-definite* if the quadratic form $\mathbf{x}^T A \mathbf{x}$ is positive semi-definite.

B.2 Some Results from Optimisation Theory

The optimisation problems which are relevant to this discussion are *minimisation problems* subject to linear constraints. The typical problem has n *decision variables* given as a vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$. The objective function f is subject to n *positivity constraints* $x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0$ and m general linear *inequality constraints*

$$\begin{array}{cccccc} b_{11}x_1 & + & b_{12}x_2 & + & \cdots & + & b_{1n}x_n & \leq & b_1 \\ b_{21}x_1 & + & b_{22}x_2 & + & \cdots & + & b_{2n}x_n & \leq & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ b_{m1}x_1 & + & b_{m2}x_2 & + & \cdots & + & b_{mn}x_n & \leq & b_m \end{array}$$

which are simply referred to as *the inequality constraints*. Although the positivity constraints are also inequality constraints, we do not include them when referring to the inequality constraints. These constraints can be specified in terms of the $m \times n$ matrix B and the vector \mathbf{b} defined as

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

respectively. The typical optimisation problem dealt with in this appendix can be stated as

Optimisation Problem B.1.

Minimise:

$$f(\mathbf{x})$$

Subject to:

$$\mathbf{B}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \geq \mathbf{0}.$$

A vector \mathbf{x} which satisfies the constraints of an optimisation problem is called a *solution* to the problem. A vector \mathbf{x} (which satisfies the constraints) for which $f(\mathbf{x})$ has its minimum value is called an *optimal solution* of Optimisation Problem B.1.

An optimisation problem of particular importance is a *linear optimisation problem*. This is an optimisation problem in which the objective function f is linear¹, thus of the form

$$f(\mathbf{x}) = c_1x_1 + c_2x_2 + \cdots + c_nx_n.$$

Define the vector $\mathbf{c} = (c_1, c_2, \dots, c_n)$. The general linear minimisation problem can be stated as

Optimisation Problem B.2 (Linear Problem).

Minimise:

$$f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$$

Subject to:

$$\mathbf{B}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \geq \mathbf{0}.$$

Linear optimisation problems can be solved using the *simplex algorithm* (see Brickman [11], Press *et al.* [27] and Saaty *et al.* [29] for example). One of the first steps in the simplex algorithm is the transformation of the inequality constraints into *equality constraints* by the introduction of *slack variables*. A maximum of m slack variables $x_{n+1}, x_{n+2}, \dots, x_{n+m}$ are required in the current problem. Define the matrix

¹The requirements for an optimisation problem to be linear are that both the objective function and the constraints must be linear. However, in all the optimisation problems studied in this work the constraints are linear by default.

$$\tilde{\mathbf{B}} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} & 1 & 0 & \cdots & 0 \\ b_{21} & b_{22} & \cdots & b_{2n} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} & 0 & 0 & \cdots & 1 \end{bmatrix}$$

and the vector $\tilde{\mathbf{x}} = (x_1, x_2, \dots, x_{n+m})$. The inequality constraints are now replaced by the equality constraints $\tilde{\mathbf{B}}\tilde{\mathbf{x}} = \mathbf{b}$.

Denote the columns of the matrix $\tilde{\mathbf{B}}$ by the column vectors $\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n+m}$:

$$\tilde{\mathbf{B}} = [\mathbf{B}_1 \quad \mathbf{B}_2 \quad \cdots \quad \mathbf{B}_{n+m}],$$

with

$$\mathbf{B}_j = \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{mj} \end{bmatrix}, \quad \mathbf{B}_{n+1} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{B}_{n+2} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{B}_{n+m} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix},$$

where $1 \leq j \leq n$. It is possible that some of the inequality constraints in Optimisation Problem B.2 are equalities instead of *real inequalities*. A *real inequality* is a statement connecting two expressions by either a *less than* relation or a *less than or equal* relation². Suppose some constraint i is an equality constraint in Optimisation Problem B.2, ie

$$b_{i1}x_1 + b_{i2}x_2 + \cdots + b_{in}x_n = b_i,$$

then the i th row of $\tilde{\mathbf{B}}$ is

$$[b_{i1} \quad b_{i2} \quad \cdots \quad b_{in} \quad 0 \quad 0 \quad \cdots \quad 0]$$

²A real inequality is different from what is mathematically known as a *strict inequality* – a statement connecting two expressions by a *less than* relation.

and \mathbf{B}_{n+i} is a null vector. Furthermore, x_{n+i} is set to zero and therefore not considered as a decision variable. Let m' be the number of inequality constraints which are real inequalities.

An important assumption³ which we make is that the constraints are linearly independent. Thus all m inequality constraints and n positivity constraints are required. Consequently the vector space \mathcal{V} , spanned by $\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n+m}$, has dimension m . There are $n + m' \leq n + m$ non-null vectors \mathbf{B}_j . All the problems that we consider are such that $m \ll n \leq n + m'$ and therefore only m of the $n + m'$ non-null vectors \mathbf{B}_j are sufficient to span the vector space \mathcal{V} . Let \mathcal{V}_B be a basis (selected solely from the set of vectors $\{\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n+m}\}$) for \mathcal{V} and let \mathcal{V}_I be the set of indices of the vectors in \mathcal{V}_B . Thus $\mathcal{V}_I = \{j \in \{1, 2, \dots, n + m\} \mid \mathbf{B}_j \in \mathcal{V}_B\}$. Any solution $\tilde{\mathbf{x}}$ to the linear optimisation problem satisfies

$$x_1 \mathbf{B}_1 + x_2 \mathbf{B}_2 + \dots + x_{n+m} \mathbf{B}_{n+m} = \mathbf{b}. \quad (37)$$

However, since \mathcal{V}_B is a basis for \mathcal{V}

$$\sum_{j \in \mathcal{V}_I} x_j^0 \mathbf{B}_j = \mathbf{b} \quad (38)$$

for some vector $\mathbf{x}_0 = (x_1^0, x_2^0, \dots, x_{n+m}^0)$ with $x_j^0 \geq 0$ for $j \in \mathcal{V}_I$ and $x_j^0 = 0$ otherwise. Hence, for each vector $\tilde{\mathbf{x}}$ satisfying (37) there exists a vector \mathbf{x}_0 satisfying (38). The following existence theorem relates $\tilde{\mathbf{x}}$ and \mathbf{x}_0 .

Theorem B.3. *Let any solution $\tilde{\mathbf{x}}$ exist for Optimisation Problem B.2. There exists a solution $\mathbf{x}_0 = (x_1^0, x_2^0, \dots, x_{n+m}^0)$ (with $x_j^0 \geq 0$ for $j \in \mathcal{V}_I$ and $x_j^0 = 0$ otherwise) such that $f(\mathbf{x}_0) = f(\tilde{\mathbf{x}})$.*

Proof. See Simonnard [30] for example. □

The vector \mathbf{x}_0 is called a *basic solution*. The simplex algorithm finds the basis \mathcal{V}_B and associated vector \mathbf{x}_0 (a basic optimal solution) such that $f(\mathbf{x}_0)$ is minimal. Theorem B.3 implies that a basic solution \mathbf{x}_0 (such that at most m elements of \mathbf{x}_0 are non-zero) is associated with every solution $\tilde{\mathbf{x}}$. In other words each solution $\tilde{\mathbf{x}}$ corresponds to a basis for \mathcal{V} (expressed as the vector \mathbf{x}_0) with the additional characteristic that $f(\mathbf{x}_0) = f(\tilde{\mathbf{x}})$. The solution process of a linear optimisation problem can be viewed as the search for an optimal basis for \mathcal{V} .

Associated with an optimisation problem is a *dual problem*. An important relationship exists between a linear optimisation problem and its dual. The dual of Optimisation Problem B.2 is

Optimisation Problem B.3 (Dual Linear Problem).

³The assumptions made in this appendix are valid for the telecommunication optimisation problems solved with flow deviation.

Maximise:

$$g(\mathbf{y}) = \mathbf{y}^T \mathbf{b}$$

Subject to:

$$\mathbf{y}^T \mathbf{B} \geq \mathbf{c}^T$$

$$\mathbf{y} \geq \mathbf{0}.$$

where $\mathbf{y} = (y_1, y_2, \dots, y_m)$. Optimisation Problem B.1 is referred to as the *primal problem*.

Theorem B.4 (Duality Theorem for Linear Programming). *If Optimisation Problem B.2 has an optimal solution \mathbf{x}_0 , then Optimisation Problem B.3 has an optimal solution \mathbf{y}_0 and $g(\mathbf{y}_0) = f(\mathbf{x}_0)$. Conversely, if Optimisation Problem B.3 has an optimal solution \mathbf{y}_0 , then Optimisation Problem B.2 has an optimal solution \mathbf{x}_0 and $f(\mathbf{x}_0) = g(\mathbf{y}_0)$.*

Proof. See Brickman [11] for example. □

This section is concluded by developing an important result for *convex optimisation problems*. We start by defining the *Lagrangian function* of an optimisation problem and its *saddle point*.

Definition B.5 (Lagrangian Function). *The Lagrangian function F of Optimisation Problem B.1 is*

$$F(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{B}\mathbf{x}, \quad (39)$$

where $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_m)$.

Definition B.6 (Saddle Point of the Lagrangian Function). *The point $(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}})$ with $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$ and $\hat{\boldsymbol{\lambda}} = (\hat{\lambda}_1, \hat{\lambda}_2, \dots, \hat{\lambda}_m)$ is called a saddle point of the Lagrangian function F (given by (39)) if*

$$F(\hat{\mathbf{x}}, \boldsymbol{\lambda}) \leq F(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}}) \leq F(\mathbf{x}, \hat{\boldsymbol{\lambda}})$$

for all \mathbf{x} and $\boldsymbol{\lambda}$.

The set \mathcal{G} defined by the intersection of the constraints is called the *feasible region* for the constraints. The following two theorems and a corollary are used to characterise the feasible region. The proofs are due to Saaty *et al.* [29].

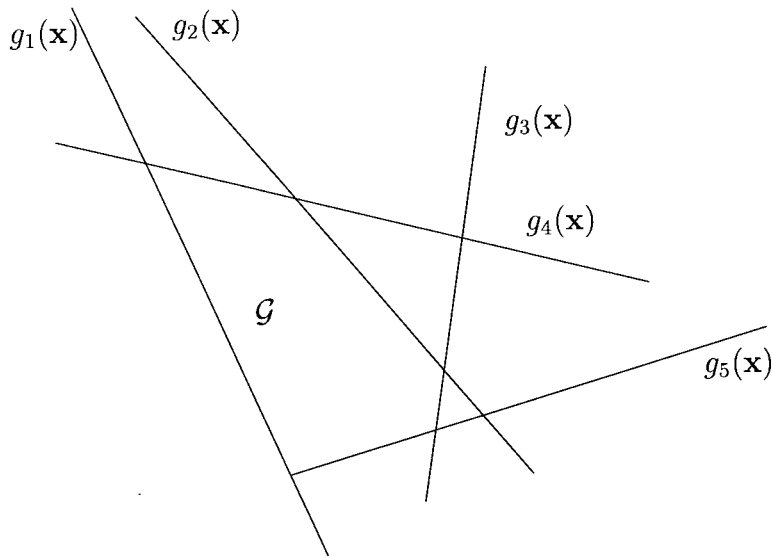


Figure B.1: Convex Feasible Region \mathcal{G} Formed by Linear Constraints

Theorem B.5. *The set \mathcal{S} of points which satisfy a constraint $g(\mathbf{x}) \leq 0$, where g is a convex function, is a convex set.*

Proof. Let \mathbf{x}_1 and \mathbf{x}_2 be two points in \mathcal{S} . Thus $g(\mathbf{x}_1) \leq 0$ and $g(\mathbf{x}_2) \leq 0$ and therefore (since g is convex)

$$g(\theta\mathbf{x}_1 + (1 - \theta)\mathbf{x}_2) \leq \theta g(\mathbf{x}_1) + (1 - \theta)g(\mathbf{x}_2) \leq 0$$

This implies that the point $\theta\mathbf{x}_1 + (1 - \theta)\mathbf{x}_2$ satisfies the constraint g and therefore $\theta\mathbf{x}_1 + (1 - \theta)\mathbf{x}_2 \in \mathcal{S}$. We conclude that \mathcal{S} is a convex set since $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}$ implies that $\theta\mathbf{x}_1 + (1 - \theta)\mathbf{x}_2 \in \mathcal{S}$. \square

Theorem B.6. *The intersection \mathcal{S} of a family of convex sets $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n$ is a convex set.*

Proof. Let \mathbf{x}_1 and \mathbf{x}_2 be two points in \mathcal{S} . Therefore $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}_k$ for $k = 1, 2, \dots, n$. \mathcal{S}_k is convex for $k = 1, 2, \dots, n$ and thus $\theta\mathbf{x}_1 + (1 - \theta)\mathbf{x}_2 \in \mathcal{S}_k$ and $\theta\mathbf{x}_1 + (1 - \theta)\mathbf{x}_2 \in \bigcap_{k=1}^n \mathcal{S}_k = \mathcal{S}$. Hence \mathcal{S} is a convex set. \square

There is an important corollary to this theorem.

Corollary B.1. *A feasible region \mathcal{G} , defined by convex constraints, is convex.*

All linear functions are convex and thus the feasible region \mathcal{G} , defined by the linear constraints $B\mathbf{x} \leq \mathbf{b}$, is convex. This is illustrated in figure B.1. The convexity of the feasible region \mathcal{G} for the constraints in Optimisation Problem B.1 is necessary for the following theorem to be applicable⁴.

Theorem B.7 (Equivalence Theorem). *Let f be convex. A necessary and sufficient condition for \mathbf{x}_0 to be a solution to Optimisation Problem B.1 is that \mathbf{x}_0 and some λ_0 comprise a saddle point of the Lagrangian function (39).*

Proof. See Saaty *et al.* [29] for the proof of a more general version of this theorem. □

We now relate this result to our current problem by means of a geometric interpretation due to Saaty *et al.* [29]. The first step is to define a *supporting hyperplane*. The following definition is taken from Fleming [17].

Definition B.7 (Supporting Hyperplane). *Let S be a closed convex set. Assume that S is neither the empty set nor E^n . A hyperplane P is called supporting for S if $P \cap S \neq \emptyset$ and S is contained in one of the two closed half-spaces bounded by P .*

In two dimensions a supporting hyperplane reduces to a *supporting line*. Figure B.2 shows a supporting line P_x passing through the point x which is contained in the convex set S . P_x satisfies the requirements of Definition B.7 since $P_x \cap S = \{x\} \neq \emptyset$ and S is contained in the upper half-space bounded by P_x .

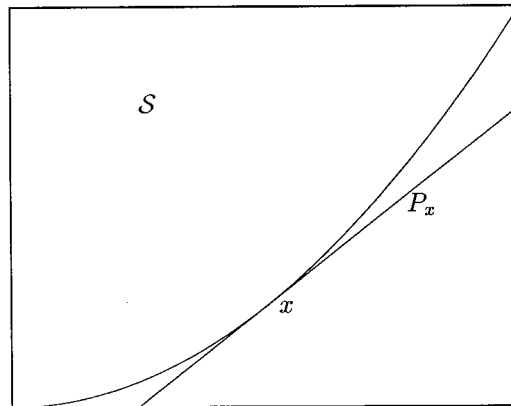
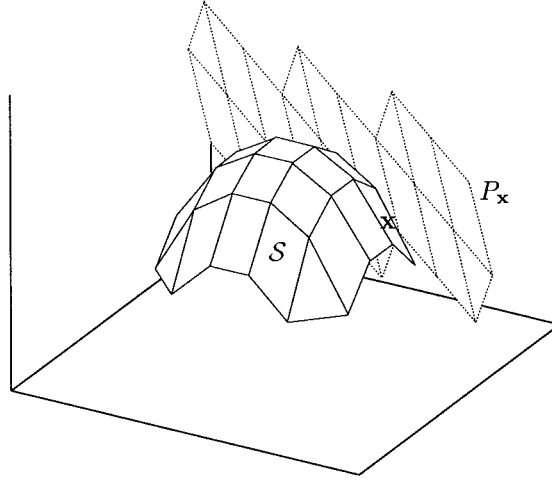


Figure B.2: A Supporting Line P_x

Figure B.3 shows a supporting hyperplane $P_{\mathbf{x}}$ passing through the point \mathbf{x} contained in the convex set S . $P_{\mathbf{x}}$ satisfies the requirements of Definition B.7 since $P_{\mathbf{x}} \cap S = \{\mathbf{x}\} \neq \emptyset$ and S is contained in the half-space bounded by $P_{\mathbf{x}}$.

⁴The convexity requirement is not stated in the formulation of Theorem B.7 since the formulation explicitly refers to Optimisation Problem B.1 whose feasible region we now know to be convex.


 Figure B.3: A Supporting Hyperplane $P_{\mathbf{x}}$

Saaty *et al.* [29] gives a geometric interpretation of the Equivalence Theorem (Theorem B.7). Let f be concave. A necessary and sufficient condition for \mathbf{x}_0 to be a solution to a maximisation problem with the same constraints as Optimisation Problem B.1, is that the normal hyperplane $H_{\mathbf{x}_0}$ (at the point \mathbf{x}_0) to the gradient vector $\nabla f(\mathbf{x}_0)$ is a supporting hyperplane of the feasible region \mathcal{G} . The normal hyperplane $H_{\mathbf{x}_0}$ has this property if \mathbf{x}_0 has its maximum projection along the outward normal $\nabla f(\mathbf{x}_0)$ to the hyperplane which implies that

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{x}_0 = \max \{ \nabla f(\mathbf{x}_0) \cdot \mathbf{w} \mid \mathbf{w} \geq \mathbf{0} \text{ and } \mathbf{B}\mathbf{w} \leq \mathbf{b} \}. \quad (40)$$

This interpretation is also valid for a convex function f in which case $-f$ is concave and

$$-\nabla f(\mathbf{x}_0) \cdot \mathbf{x}_0 = \max \{ -\nabla f(\mathbf{x}_0) \cdot \mathbf{w} \mid \mathbf{w} \geq \mathbf{0} \text{ and } \mathbf{B}\mathbf{w} \leq \mathbf{b} \},$$

therefore

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{x}_0 = \min \{ \nabla f(\mathbf{x}_0) \cdot \mathbf{w} \mid \mathbf{w} \geq \mathbf{0} \text{ and } \mathbf{B}\mathbf{w} \leq \mathbf{b} \}.$$

We consider an example (due to Saaty *et al.* [29]) of the maximisation of a concave quadratic function.

Optimisation Problem B.4.

Maximise:

$$f(\mathbf{x}) = 6x_1 - 2x_1^2 + 2x_1x_2 - 2x_2^2$$

Subject to:

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ \mathbf{x} &\geq \mathbf{0}. \end{aligned}$$

The gradient vector is $\nabla f(\mathbf{x}) = (6 - 4x_1 + 2x_2, 2x_1 - 4x_2)$. We want to determine whether or not the solution $\mathbf{x} = (1, 1)$ is optimal. $\nabla f(\mathbf{x}) = (4, -2)$ and $\nabla f(\mathbf{x}) \cdot \mathbf{x} = 2$ at this point. Let $\mathbf{w} = (3/2, 1/2)$, then $\nabla f(\mathbf{x}) \cdot \mathbf{w} = 5$. Thus $\nabla f(\mathbf{x}) \cdot \mathbf{x} < \nabla f(\mathbf{x}) \cdot \mathbf{w}$ and consequently \mathbf{x} is not an optimal point since (40) does not hold. This situation is illustrated in figure B.4. It is geometrically obvious from the figure that \mathbf{x} is not optimal since the acute angle between \mathbf{x} and $\nabla f(\mathbf{x})$ is larger than the acute angle between \mathbf{w} and $\nabla f(\mathbf{x})$.

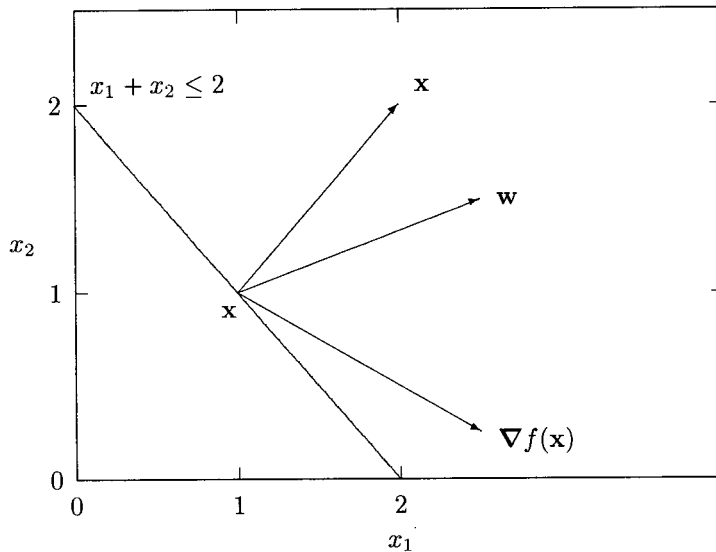
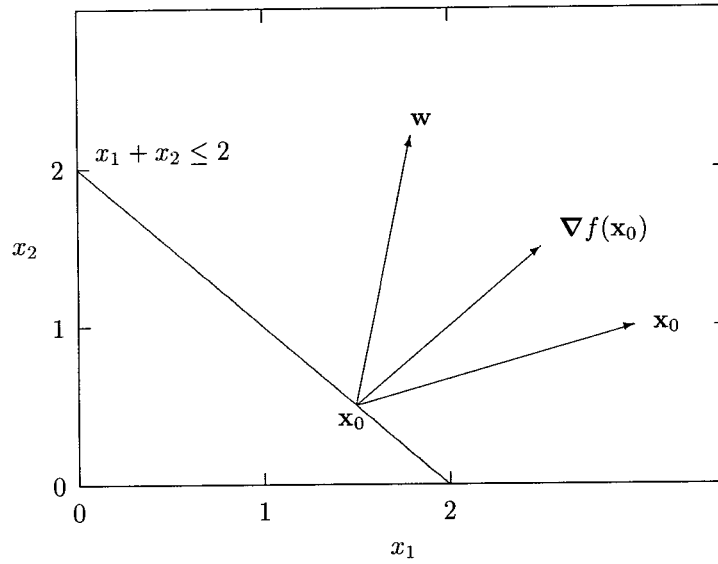


Figure B.4: A Non-Optimal Point \mathbf{x}

Consider the point $\mathbf{x}_0 = (3/2, 1/2)$ where $\nabla f(\mathbf{x}_0) = (1, 1)$ and $\nabla f(\mathbf{x}_0) \cdot \mathbf{x}_0 = 2$. Saaty *et al.* [29] give \mathbf{x}_0 as the optimal point since $\nabla f(\mathbf{x}_0) \cdot \mathbf{x}_0 \geq \nabla f(\mathbf{x}_0) \cdot \mathbf{w}$ for any other point \mathbf{w} which satisfies the constraints. Let $\mathbf{w} = (0.3, 1.7)$ for example, then $\nabla f(\mathbf{x}_0) \cdot \mathbf{w} = 2$. Thus $\nabla f(\mathbf{x}_0) \cdot \mathbf{x} = \nabla f(\mathbf{x}_0) \cdot \mathbf{w} = 2$. This situation is illustrated in figure B.5.


 Figure B.5: An Optimal Point \mathbf{x}_0

B.3 The Frank-Wolfe Method

The ideas developed in the previous two sections are now applied to the optimisation technique on which flow deviation is based.

Let \mathbf{A} and \mathbf{B} be the matrices

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

and \mathbf{x} , \mathbf{a} and \mathbf{b} the column vectors

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

respectively. Additionally, \mathbf{A} is symmetric and positive semi-definite. The Frank-Wolfe method (see Boot [10], Künzi *et al.* [24] and Saaty *et al.* [29] for example) is designed to solve the following

quadratic optimisation problem.

Optimisation Problem B.5.

Minimise:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{a}^T \mathbf{x}$$

Subject to:

$$\mathbf{B} \mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \geq \mathbf{0}$$

Denote the gradient vector of f at the point \mathbf{x} by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix} = \nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} - \mathbf{a},$$

where the result of Theorem B.2 has been used. The Equivalence Theorem (Theorem B.7) implies that \mathbf{x}_0 is an optimal solution of Optimisation Problem B.5 if and only if

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{x}_0 = \min \{ \nabla f(\mathbf{x}_0) \cdot \mathbf{w} \mid \mathbf{w} \geq \mathbf{0} \text{ and } \mathbf{B}\mathbf{w} \leq \mathbf{b} \}$$

Let $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0) = \nabla f(\mathbf{x}_0)$, then the previous equation (in matrix form) becomes

$$\mathbf{f}_0^T \mathbf{x}_0 = \min \left\{ \mathbf{f}_0^T \mathbf{w} \mid \mathbf{w} \geq \mathbf{0} \text{ and } \mathbf{B}\mathbf{w} \leq \mathbf{b} \right\}$$

and by the Duality Theorem (Theorem B.4)

$$\mathbf{f}_0^T \mathbf{x}_0 = \max \left\{ \mathbf{u}^T \mathbf{b} \mid \mathbf{u} \geq \mathbf{0} \text{ and } \mathbf{u}^T \mathbf{B} \geq \mathbf{f}_0^T \right\}$$

where $\mathbf{w} = (w_1, w_2, \dots, w_n)$ and $\mathbf{u} = (u_1, u_2, \dots, u_m)$. Since

$$\mathbf{f}_0^T \mathbf{x}_0 - \max \left\{ \mathbf{u}^T \mathbf{b} \mid \mathbf{u} \geq \mathbf{0} \text{ and } \mathbf{u}^T \mathbf{B} \geq \mathbf{f}_0^T \right\} = 0$$

it follows that

$$\min \left\{ \mathbf{f}_0^T \mathbf{x}_0 - \mathbf{u}^T \mathbf{b} \mid \mathbf{u} \geq \mathbf{0} \text{ and } \mathbf{u}^T \mathbf{B} \geq \mathbf{f}_0^T \right\} = 0.$$

Define

$$F(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x})^T \mathbf{x} - \mathbf{u}^T \mathbf{b},$$

then

$$\begin{aligned} F(\mathbf{x}, \mathbf{u}) &= \mathbf{f}(\mathbf{x})^T \mathbf{x} - \mathbf{u}^T \mathbf{b} \\ &= (2\mathbf{A}\mathbf{x} - \mathbf{a})^T \mathbf{x} - \mathbf{u}^T \mathbf{b} \\ &= (2\mathbf{x}^T \mathbf{A} - \mathbf{a}^T) \mathbf{x} - \mathbf{u}^T \mathbf{b} \\ &= 2\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{a}^T \mathbf{x} - \mathbf{u}^T \mathbf{b}, \end{aligned}$$

where the symmetry of \mathbf{A} has been used. Thus at an optimal value of f , \mathbf{x} and \mathbf{u} can be found such that

$$\mathbf{x} \geq \mathbf{0}, \quad \mathbf{u} \geq \mathbf{0}, \quad \mathbf{B}\mathbf{x} \leq \mathbf{b}, \quad \mathbf{f}(\mathbf{x})^T \leq \mathbf{u}^T \mathbf{B} \quad \text{and} \quad F(\mathbf{x}, \mathbf{u}) = 0. \quad (41)$$

We introduce slack variables in the form of the vectors $\mathbf{v} = (v_1, v_2, \dots, v_n)$ and $\mathbf{y} = (y_1, y_2, \dots, y_m)$. The first set of inequality constraints in (41) can now be stated as

$$\mathbf{B}\mathbf{x} + \mathbf{y} = \mathbf{b} \quad (42)$$

and the second set (after applying the transpose operation on both sides) as

$$\begin{aligned} \mathbf{0} &\leq \mathbf{B}^T \mathbf{u} - \mathbf{f}(\mathbf{x}) \\ &= \mathbf{B}^T \mathbf{u} - 2\mathbf{A}\mathbf{x} + \mathbf{a} \\ -\mathbf{a} &\leq \mathbf{B}^T \mathbf{u} - 2\mathbf{A}\mathbf{x}. \end{aligned}$$

Subtraction of the slack variables \mathbf{v} yields

$$-\mathbf{a} = \mathbf{B}^T \mathbf{u} - 2\mathbf{A}\mathbf{x} - \mathbf{v}.$$

The above equation can be used to rewrite $F(\mathbf{x}, \mathbf{u})$ as

$$\begin{aligned} F(\mathbf{x}, \mathbf{u}) &= 2\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{a}^T \mathbf{x} - \mathbf{u}^T \mathbf{b} \\ &= 2\mathbf{x}^T \mathbf{A}\mathbf{x} + (\mathbf{B}^T \mathbf{u} - 2\mathbf{A}\mathbf{x} - \mathbf{v})^T \mathbf{x} - \mathbf{u}^T \mathbf{b} \\ &= 2\mathbf{x}^T \mathbf{A}\mathbf{x} + (\mathbf{u}^T \mathbf{B} - 2\mathbf{x}^T \mathbf{A} - \mathbf{v}^T) \mathbf{x} - \mathbf{u}^T \mathbf{b} \\ &= 2\mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{u}^T \mathbf{B}\mathbf{x} - 2\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{v}^T \mathbf{x} - \mathbf{u}^T \mathbf{b} \\ &= -\mathbf{v}^T \mathbf{x} + \mathbf{u}^T (\mathbf{B}\mathbf{x} - \mathbf{b}) \\ &= -\mathbf{v}^T \mathbf{x} - \mathbf{u}^T \mathbf{y} \end{aligned}$$

by using (42) and the symmetry of \mathbf{A} . Therefore the equality $F(\mathbf{x}, \mathbf{u}) = 0$ in (41) becomes

$$\mathbf{v}^T \mathbf{x} + \mathbf{u}^T \mathbf{y} = 0.$$

Define the $2(n + m)$ -dimensional vector \mathbf{w} as

$$\mathbf{w} = \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{y} \\ \mathbf{v} \end{bmatrix},$$

then F can be considered as a function of \mathbf{w} and the inequalities and equality in (41) are given by

$$\mathbf{B}\mathbf{x} + \mathbf{y} = \mathbf{b} \tag{43}$$

$$\mathbf{B}^T \mathbf{u} - 2\mathbf{A}\mathbf{x} - \mathbf{v} = -\mathbf{a} \tag{44}$$

$$\mathbf{v}^T \mathbf{x} + \mathbf{u}^T \mathbf{y} = 0 \tag{45}$$

$$\mathbf{w} \geq \mathbf{0}. \tag{46}$$

Constraints (43) and (44) can be written as a single matrix equation $\mathbf{D}\mathbf{w} = \mathbf{d}$ where \mathbf{D} and \mathbf{d} are defined as

$$D = \begin{bmatrix} B & 0_m & I_m & 0_n \\ -2A & B^T & 0_m & -I_n \end{bmatrix} \quad \text{and} \quad \mathbf{d} = \begin{bmatrix} \mathbf{b} \\ -\mathbf{a} \end{bmatrix}$$

respectively. Note that D is an $(m+n) \times 2(m+n)$ matrix and \mathbf{d} an $(m+n)$ -dimensional vector. (45) is a requirement for an optimal point and (46) represents the positivity constraints. Optimisation Problem B.5 can be stated as

Optimisation Problem B.6.

Minimise:

$$F(\mathbf{w}) = -\mathbf{v}^T \mathbf{x} - \mathbf{u}^T \mathbf{y}$$

Subject to:

$$D\mathbf{w} = \mathbf{d}$$

$$\mathbf{w} \geq \mathbf{0}.$$

We now discuss the Frank-Wolfe algorithm which is designed to solve this optimisation problem. Let $\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_{2(m+n)}$ be the column vectors of D and let \mathcal{V} be the vector space spanned by these column vectors. The Frank-Wolfe algorithm is an iterative algorithm and at the start of iteration k , we have a solution $\mathbf{w}^k = (w_1^k, w_2^k, \dots, w_{2(m+n)}^k) = (\mathbf{x}^k, \mathbf{u}^k, \mathbf{y}^k, \mathbf{v}^k)$ such that

$$\sum_{j \in \mathcal{V}_I^k} w_j^k \mathbf{D}_j = \mathbf{d}.$$

The first step is to calculate $F(\mathbf{w}^k)$. $F(\mathbf{w}^k)$ is non-negative since the minimum of F is known to be 0 (from (41) and (45)). If $F(\mathbf{w}^k) = 0$, \mathbf{w}^k is an optimal point and the algorithm terminates. If $F(\mathbf{w}^k) > 0$ the optimal point has not yet been reached. A new point (solution) \mathbf{w}^{k+1} is now constructed from \mathbf{w}^k by using the simplex algorithm to solve a linear optimisation problem. Calculate the gradient vector $\nabla F(\mathbf{w}^k)$ at the current point and form the following linear optimisation problem.

Optimisation Problem B.7.

Minimise:

$$\tilde{F}(\varphi) = [\nabla F(\mathbf{w}^k)]^T \varphi$$

Subject to:

$$\begin{aligned} \mathbf{D}\boldsymbol{\varphi} &= \mathbf{d} \\ \boldsymbol{\varphi} &\geq \mathbf{0}, \end{aligned}$$

where $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \dots, \varphi_{2(n+m)})$ represents the decision variables.

Apply the simplex algorithm to this linear problem to obtain a basis \mathcal{V}_D for the vector space \mathcal{V} spanned by the vectors $\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_{2(n+m)}$. As stated in the description of the simplex algorithm in section B.2, the basis \mathcal{V}_D is chosen from the vectors $\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_{2(n+m)}$ and \mathcal{V}_I is the set of indices of the vectors \mathbf{D}_j in \mathcal{V}_D . The basis has an associated solution $\boldsymbol{\varphi}$. Note that \mathcal{V}_D is an optimal basis for Optimisation Problem B.7 and not necessarily for Optimisation Problems B.5 and B.6.

If $F(\boldsymbol{\varphi}) < F(\mathbf{w}^k)$, then set $\mathbf{w}^{k+1} = \boldsymbol{\varphi}$. Otherwise, perform a line search to find the $\alpha \in [0, 1]$ such that $F((1 - \alpha)\mathbf{w}^k + \alpha\boldsymbol{\varphi})$ is minimised and set $\mathbf{w}^{k+1} = (1 - \alpha)\mathbf{w}^k + \alpha\boldsymbol{\varphi}$.

The complete Frank-Wolfe algorithm can now be stated as follows.

Algorithm B.1 (Frank-Wolfe). *Start with an initial solution \mathbf{w}^1 .*

1. $k \leftarrow 1$
2. **if** $F(\mathbf{w}^k) = 0$ **then**
 the algorithm terminates
 else
 continue
 endif
3. Calculate the gradient vector $\nabla F(\mathbf{w}^k)$ at the current point \mathbf{w}^k .
4. Form Optimisation Problem B.7 and obtain an optimal solution $\boldsymbol{\varphi}$ by means of the simplex algorithm.
5. **if** $F(\boldsymbol{\varphi}) < F(\mathbf{w}^k)$ **then**
 $\mathbf{w}^{k+1} \leftarrow \boldsymbol{\varphi}$
 $k \leftarrow k + 1$ and **return** to step 2
 else
 continue
 endif
6. (a) Perform a line search to find the $\alpha \in [0, 1]$ such that $F((1 - \alpha)\mathbf{w}^k + \alpha\boldsymbol{\varphi})$ is minimised
 (b) $\mathbf{w}^{k+1} \leftarrow (1 - \alpha)\mathbf{w}^k + \alpha\boldsymbol{\varphi}$.
7. $k \leftarrow k + 1$ and **return** to step 2.

If the algorithm terminates during iteration $k = K$, set $\mathbf{x}_0 = \mathbf{w}^K$ which is an optimal solution to Optimisation Problem B.5.

Although the Frank-Wolfe method was originally designed to minimise a convex quadratic function $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{a}^T \mathbf{x}$, it can be adapted to minimise a general convex function. The reason is that the Equivalence Theorem (Theorem B.7) holds for any convex function. Thus the requirement for a point \mathbf{x}_0 to be optimal is still

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{x}_0 = \min \{ \nabla f(\mathbf{x}_0) \cdot \mathbf{w} \mid \mathbf{w} \geq \mathbf{0} \text{ and } \mathbf{B}\mathbf{w} \leq \mathbf{b} \},$$

provided that f is convex. However, $F(\mathbf{x}, \mathbf{u})$ has a more general form and the algorithm is not stated in terms of F but in terms of the original objective function f . The objective function \tilde{F} in Optimisation Problem B.7 is replaced with

$$\tilde{f}(\varphi) = [\nabla f(\mathbf{w}^k)]^T \varphi. \quad (47)$$

We present the following algorithm to solve Optimisation Problem B.5 for a general convex function f .

Algorithm B.2 (The Modified Frank-Wolfe Algorithm). *Start with an initial solution \mathbf{w}^1 .*

1. $k \leftarrow 1$
2. Calculate the gradient vector $\nabla f(\mathbf{w}^k)$ at the current point \mathbf{w}^k .
3. Form optimisation problem B.7 (with \tilde{F} replaced by \tilde{f} given by (47)) and obtain an optimal solution φ by means of the simplex algorithm.
4. (a) Perform a line search to find the $\alpha \in [0, 1]$ such that $f((1 - \alpha)\mathbf{w}^k + \alpha\varphi)$ is minimised
 (b) $\mathbf{w}^{k+1} \leftarrow (1 - \alpha)\mathbf{w}^k + \alpha\varphi$.
5. **if** the termination criterion has been met **then**
 the algorithm terminates
else
 $k \leftarrow k + 1$ and **return** to step 2
endif

An example of a simple termination criterion is to test whether

$$|F(\mathbf{w}^{k+1}) - F(\mathbf{w}^k)| < \varepsilon$$

where $\varepsilon > 0$ is some convergence criterion.

One of the only differences between the original Frank-Wolfe algorithm and the modified version is that a line search is performed regardless of whether or not $f(\varphi) < f(\mathbf{w}^k)$.

Appendix C

The Fibonacci Search Method

This appendix presents a specific *line search* method. A line search is a particular one-dimensional optimisation problem which can be described as follows. Consider a vector-valued function $f : \mathcal{D} \mapsto \mathbb{R}$ with $\mathcal{D} \subseteq \mathbb{R}^n$. Let \mathbf{x} and \mathbf{y} be two points in \mathcal{D} and let $\mathcal{E} = \{\alpha \in \mathbb{R} \mid \mathbf{x} + \alpha\mathbf{y} \in \mathcal{D}\}$. Define the function $g : \mathcal{E} \mapsto \mathbb{R}$ as $g(\alpha) = f(\mathbf{x} + \alpha\mathbf{y})$. A line search method finds the value of $\alpha \in [a_1, b_1] \subseteq \mathcal{E}$ which minimises g . The *Fibonacci search* method is a specific line search method which is applicable when g is unimodal. The following definition is due to Foulds [18].

Definition C.1 (Unimodal Function). *Let the function $h : \mathcal{H} \mapsto \mathbb{R}$ have a minimum at the point c in the interval $[a, b] \subseteq \mathcal{H} \subset \mathbb{R}$. h is unimodal if $h(x)$ is strictly decreasing for $x < c$ and $h(x)$ is strictly increasing for $x > c$.*

The Fibonacci search method is based on the *Fibonacci numbers* defined as

$$\begin{aligned} A_0 &= 0 \\ A_1 &= 1 \\ A_k &= A_{k-1} + A_{k-2} \quad \text{for } k = 2, 3, \dots \end{aligned}$$

Let the search interval be $[a_1, b_1]$. The Fibonacci method reduces this interval in $M - 1$ iterations to the interval $[a_M, b_M]$. M has to be chosen in advance. A new interval $[a_{m+1}, b_{m+1}]$ is constructed from the current interval $[a_m, b_m]$ during iteration m . Define the numbers s_m and t_m as

$$\begin{aligned} s_m &= a_m + (b_m - a_m) \frac{A_{M-m}}{A_{M+2-m}} \\ t_m &= a_m + (b_m - a_m) \frac{A_{M+1-m}}{A_{M+2-m}} \end{aligned}$$

for $m = 1, 2, \dots, M - 1$. Note that $a_m < s_m \leq t_m < b_m$ and therefore s_m and t_m are points in $[a_m, b_m]$. The interval $[a_{m+1}, b_{m+1}]$ is constructed, based on the relation between $g(s_m)$ and $g(t_m)$. There are three possibilities:

- if $g(s_m) < g(t_m)$ then set $a_{m+1} = a_m$ and $b_{m+1} = t_m$
- if $g(s_m) > g(t_m)$ then set $a_{m+1} = s_m$ and $b_{m+1} = b_m$
- if $g(s_m) = g(t_m)$ then a new Fibonacci search is performed on the interval $[s_m, t_m]$ (possibly) with a new value of M .

In iteration $M - 1$, s_{M-1} is

$$\begin{aligned} s_{M-1} &= a_{M-1} + (b_{M-1} - a_{M-1}) \frac{A_1}{A_3} \\ &= \frac{1}{2}(a_{M-1} + b_{M-1}) \\ &= a_{M-1} + (b_{M-1} - a_{M-1}) \frac{A_2}{A_3} \\ &= t_{M-1}. \end{aligned}$$

However, the minimum of g is not necessarily at $s_{M-1} = t_{M-1}$ since s_{M-1} is merely the midpoint of the current interval $[a_{M-1}, b_{M-1}]$. Therefore we set $t_{M-1} = s_{M-1} + \varepsilon$, where ε is some small length. This forces at least one further iteration.

The object of a Fibonacci search is the reduction of the interval $[a_1, b_1]$ to the interval $[a_M, b_M]$, where the ratio between the lengths of $[a_1, b_1]$ and $[a_M, b_M]$ is at least r . Therefore M and ε have to be selected such that the maximum possible length of $[a_M, b_M]$ is not larger than $(b_1 - a_1)/r$. The maximum length of the interval $[a_M, b_M]$ is obtained when $g(s_{M-1}) < g(t_{M-1})$ and is

$$\begin{aligned} b_M - a_M &= t_{M-1} - a_{M-1} \\ &= \left[\frac{1}{2}(a_{M-1} + b_{M-1}) + \varepsilon \right] - a_{M-1} \\ &= \frac{1}{2}(b_{M-1} - a_{M-1}) + \varepsilon. \end{aligned}$$

Therefore

$$\begin{aligned} 2[(b_M - a_M) - \varepsilon] &= b_{M-1} - a_{M-1} \\ &= \frac{b_1 - a_1}{\frac{3}{2}A_{M-2} + A_{M-3}}, \end{aligned}$$

where the last equality is proved by Foulds [18]. Thus ε and M have to be chosen such that

$$\begin{aligned} r &\leq \frac{b_1 - a_1}{b_M - a_M} \\ \frac{1}{r} &\geq \frac{b_M - a_M}{b_1 - a_1} \\ &= \frac{\varepsilon}{b_1 - a_1} + \frac{1}{3A_{M-2} + 2A_{M-3}}. \end{aligned}$$

Appendix D

Newton's Method

This appendix presents an optimisation method based on the *Newton-Raphson method*. The classical Newton-Raphson method is reviewed in section D.1 and an optimisation method based on it is discussed in sections D.2 and D.3.

D.1 The Newton-Raphson Method

Consider a real-valued, differentiable function $f : \mathcal{D} \mapsto \mathbb{R}$ with $\mathcal{D} \subseteq \mathbb{R}$ and $f(x_0) = 0$ for some (still unknown) $x_0 \in \mathcal{D}$. The classical Newton-Raphson method (see Maron *et al.* [26], Press *et al.* [27] and Saaty *et al.* [29], for example) is an iterative method for finding x_0 in the interval \mathcal{D} . This method starts with an initial estimate x^0 of x_0 and for each $k \geq 0$, the next estimate x^{k+1} is obtained from x^k by the *iterative step*:

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}. \quad (48)$$

The Newton-Raphson method can also be extended to the multi-dimensional case (see Press *et al.* [27], for example), where $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and the system of n equations in n variables

$$g_1(\mathbf{x}) = 0, g_2(\mathbf{x}) = 0, \dots, g_n(\mathbf{x}) = 0$$

has to be solved. Each function g_ℓ can be expanded as a *Taylor series* to obtain

$$g_\ell(\mathbf{x}^k + \Delta \mathbf{x}^k) = g_\ell(\mathbf{x}^k) + \sum_{j=1}^n \frac{\partial}{\partial x_j} g_\ell(\mathbf{x}^k) \Delta x_j^k + O(\Delta \mathbf{x}^k),$$

where $\Delta \mathbf{x}^k = (\Delta x_1^k, \Delta x_2^k, \dots, \Delta x_n^k)$ and $O(\Delta \mathbf{x}^k)$ represents the sum of the quadratic and higher terms. This can be written in matrix form as

$$g_\ell(\mathbf{x}^k + \Delta \mathbf{x}^k) = g_\ell(\mathbf{x}^k) + [\nabla g_\ell(\mathbf{x}^k)]^T \Delta \mathbf{x}^k + O(\Delta \mathbf{x}^k), \quad (49)$$

where the matrix representation of the gradient vector $\nabla g_\ell(\mathbf{x}^k)$ is a column vector as are the matrix representations of all other vectors used in this appendix.

Define the vector $\mathbf{g}(\mathbf{x})$ and the *Jacobian matrix* \mathbf{J} as

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_n(\mathbf{x}) \end{bmatrix} \quad \text{and} \quad \mathbf{J} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \dots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \dots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \dots & \frac{\partial g_n}{\partial x_n} \end{bmatrix}$$

respectively. The following matrix equation describes the entire system of equations (49):

$$\mathbf{g}(\mathbf{x}^k + \Delta \mathbf{x}^k) = \mathbf{g}(\mathbf{x}^k) + \mathbf{J} \Delta \mathbf{x}^k + O(\Delta \mathbf{x}^k),$$

where, with a slight abuse of notation, $O(\Delta \mathbf{x}^k)$ is now a column vector with the ℓ th entry being the sum of the quadratic and higher terms of the function g_ℓ . Ignoring these non-linear terms yields

$$\mathbf{g}(\mathbf{x}^k + \Delta \mathbf{x}^k) \approx \mathbf{g}(\mathbf{x}^k) + \mathbf{J} \Delta \mathbf{x}^k.$$

Finally, by setting $\mathbf{g}(\mathbf{x}^k + \Delta \mathbf{x}^k) = \mathbf{0}$, one obtains

$$\mathbf{J} \Delta \mathbf{x}^k = -\mathbf{g}(\mathbf{x}^k) \quad (50)$$

and therefore the multi-dimensional equivalent of (48) is

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta \mathbf{x}^k. \quad (51)$$

Press *et al.* [27] suggest solving (50) by means of *LU decomposition* to obtain $\Delta \mathbf{x}^k$.

D.2 A Cauchy-Type Steepest Descend Method

This section discusses a method of unconstrained minimisation based on the Newton-Raphson method. The derivation presented here is due to Himmelblau [19]. Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and let $F : \mathcal{E} \mapsto \mathbb{R}$ be a differentiable, convex function with $\mathcal{E} \subseteq \mathbb{R}^n$. The object of the minimisation method is to find the minimum of F by means of a *Cauchy-type steepest descent method*. Consider the main iteration of a general Cauchy-type steepest descent method:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta \mathbf{x}^k. \quad (52)$$

The vector $\Delta \mathbf{x}^k$ (the *increment*) is such that for each iteration k , the value of F at the next point \mathbf{x}^{k+1} is smaller than $F(\mathbf{x}^k)$. Thus, as the term steepest descent indicates, the algorithm moves closer to the minimum of F during each iteration. The global minimum of F is at the critical point \mathbf{x} (where $\nabla F(\mathbf{x}) = \mathbf{0}$) since F is convex (see appendix A). Therefore the point \mathbf{x} satisfies the system of equations

$$\begin{aligned} \frac{\partial}{\partial x_1} F(\mathbf{x}) &= 0 \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) &= 0 \\ &\vdots \\ \frac{\partial}{\partial x_n} F(\mathbf{x}) &= 0. \end{aligned}$$

Note that the iterative step (52) of the steepest descent method is the same as the iterative step (51) of the Newton-Raphson method. The above system of equations can be solved using the Newton-Raphson method and the increment $\Delta \mathbf{x}^k$ calculated by the Newton-Raphson method can be substituted into (52). Thus, the minimisation problem is transformed into a root finding problem. The increment calculated by the Newton-Raphson method is appropriate for the steepest descent method since both methods move towards the same point \mathbf{x} (where $\nabla F(\mathbf{x}) = \mathbf{0}$). In other words, the root \mathbf{x} of $\nabla F(\mathbf{x}) = \mathbf{0}$ (sought by the Newton-Raphson method) corresponds to the point \mathbf{x} where F has its minimum (sought by the steepest descent method).

In order to apply the Newton-Raphson method (as described in section D.1), let

$$g_\ell(\mathbf{x}) = \frac{\partial}{\partial x_\ell} F(\mathbf{x}).$$

Therefore, (50) becomes

$$\mathbf{H}\Delta\mathbf{x}^k = -\nabla F(\mathbf{x}^k)$$

where \mathbf{H} is the *Hessian matrix*

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}.$$

Thus

$$\Delta\mathbf{x}^k = -\mathbf{H}^{-1}\nabla F(\mathbf{x}^k)$$

which can be substituted into (52) to yield the iterative step of this steepest descent method

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{H}^{-1}\nabla F(\mathbf{x}^k). \quad (53)$$

D.3 A Gradient Projection Method

The unconstrained optimisation method discussed in the previous section can be modified to solve constrained optimisation problems by applying the technique of *gradient projection* (see Himmelblau [19] and Jacoby *et al.* [20] for discussions of the general method). Consider the following minimisation problem

Optimisation Problem D.1.

Minimise:

$$h(\mathbf{x})$$

Subject to:

$$\mathbf{x} \geq \mathbf{0}$$

where h is differentiable and convex and the only constraints are positivity constraints (see appendix B). Applying a simple gradient projection method (see Bertsekas *et al.* [9]) transforms the iterative step (53) in the Cauchy-type steepest descent method into

$$\mathbf{x}^{k+1} = \max \{0, \mathbf{x}^k - \mathbf{H}^{-1} \nabla h(\mathbf{x}^k)\}.$$

Thus, the result is projected onto the positive ortant. Jacoby *et al.* [20] provide reasons for the convergence of this technique.

Bertsekas *et al.* [9] point out that the Hessian matrix \mathbf{H} is often approximated with the diagonal matrix

$$\begin{bmatrix} \frac{\partial^2 h}{\partial x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 h}{\partial x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 h}{\partial x_n^2} \end{bmatrix}.$$

The iterative step is then given by

$$x_\ell^{k+1} = \max \left\{ 0, x_\ell^k - \left[\frac{\partial^2}{\partial x_\ell^2} h(\mathbf{x}^k) \right]^{-1} \frac{\partial}{\partial x_\ell} h(\mathbf{x}^k) \right\} \quad (54)$$

for $\ell = 1, 2, \dots, n$.

Consequently, if a constrained optimisation problem (involving a differentiable, convex objective function) can be rewritten in the form of optimisation problem D.1, the method of section D.2 can be used to solve it. Chapter 2 deals with a constrained optimisation problem to which this technique is applied.

Bibliography

- [1] G Ash *et al.*. Traffic Engineering and QoS Methods for IP-, ATM- and TDM-Based Networks. draft-ietf-tewg-qos-routing-01.txt, March 2001. Obtainable from <http://www.ietf.org/ietf/lid-abstracts.txt>.
- [2] N Anerousis and AA Lazar. Virtual Path Control for ATM Networks with Call Level Quality of Service Quarantees. *IEEE ACM Transactions on Networking* **6:2**, pp 222-236, April 1998.
- [3] H Anton. *Elementary Linear Algebra*. John Wiley & Sons, New York, 1994.
- [4] ÅArvidsson. High Level B-ISDN/ATM Traffic Management in Real Time, *Performance Modelling and Evaluation of ATM Networks*, **1:**, pp. 177-207. Chapman & Hall, London, 1995.
- [5] Å Arvidsson, SA Berezner and AE Krzesinski. Virtual Path Connection Management in ATM Networks. Proceedings *6th IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, Bradford, UK, July 1998.
- [6] JE Burns, TJ Ott, JM de Kock and AE Krzesinski. Path Selection and Bandwidth Allocation in MPLS Networks: a Non-linear Programming Approach. Proceedings *ITCom 2001*, Denver, Colorado, USA, August 2001.
- [7] ÅArvidsson, SA Berezner and AE Krzesinski. The Design and Management of ATM Virtual Path Connection Networks. Proceedings *7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '99)*, College Park, Maryland, October 1999, pp 1-9.
- [8] SA Berezner and AE Krzesinski. Call Admission and Routing in ATM Networks Based on Virtual Path Separation. IFIP TC6/WG6.2 Proceedings *4th International Conference on Broadband Communications*, Stuttgart, Germany, April 1998. Chapman & Hall (Eds PJ Kühn and R Ulrich), pp 461-472.
- [9] D Bertsekas and R Gallager. *Data Networks*. Prentice-Hall International, Englewood Cliffs, 1992.
- [10] JCG Boot. *Quadratic Programming: Algorithms, Anomalies, Applications*. North-Holland Publishing Company, Amsterdam, 1964.

- [11] L Brickman. *Mathematical Introduction to Linear Programming and Game Theory*. Springer-Verlag, New York, 1989.
- [12] RB Cooper *Introduction to Queueing Theory*. Edward Arnold, London, 1981.
- [13] B Davie, P Coolan and Y Rekhter. *Switching in IP Networks: IP Switching, Tag Switching and Related Technologies*. Morgan Kaufmann Publishers, San Francisco, 1998.
- [14] JM de Kock and AE Krzesinski. The Design of Optimal Virtual Path Connection Networks with Service Separation. *Proceedings of the 2nd International Conference on Information, Communications and Signal Processing*, Singapore, December 1999.
- [15] A Faragó, S Blaabjerg, L Ast, G Gordos and T Henk. A New Degree of Freedom in ATM Network Dimensioning: Optimizing the Logical Configuration. *IEEE Journal on Selected Areas in Communications*, **13:7**, pp 1199-1206, September 1995.
- [16] E Fischer. *Intermediate Real Analysis*. Springer-Verlag, New York, 1983.
- [17] WH Fleming. *Functions of Several Variables*. Springer-Verlag, New York, 1977.
- [18] LR Foulds. *Optimization Techniques*. Springer-Verlag, New York, 1981.
- [19] DM Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, Inc., New York, 1972.
- [20] SLS Jacoby, JS Kowalik and JT Pizzo. *Iterative Methods for Nonlinear Optimization Problems*. Prentice-Hall, Englewood Cliffs, 1972.
- [21] A Kerschenbaum. *Telecommunication Design Algorithms*. McGraw-Hill, New York, 1993.
- [22] L Kleinrock. *Queueing System*, vol. 1: Theory. John Wiley & Sons, New York, 1975.
- [23] L Kleinrock. *Queueing System*, vol. 2: Computer Applications. John Wiley & Sons, New York, 1976.
- [24] HP Künzi and W Krelle. *Nichtlineare Programmierung*. Springer-Verlag, Berlin, 1962.
- [25] U Manber. *Introduction to Algorithms*. Addison-Wesley, Reading, Massachusetts, 1989.
- [26] MJ Maron and RJ Lopez. *Numerical Analysis: a Practical Approach*. Wadsworth Publishing Company, Belmont, California, 1991.
- [27] WH Press, SA Teukolsky, WT Vetterling and BP Flannery. *Numerical Recipes in C* Second Edition. Cambridge University Press, 1992.
- [28] KW Ross. *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer-Verlag, Berlin, 1995.
- [29] TL Saaty and J Bram. *Nonlinear Mathematics*. Dover Publications, New York, 1981.
- [30] M Simonnard. *Linear Programming*. Prentice-Hall, Englewood Cliffs, 1966.
- [31] A Tucker. *Applied Combinatorics* Third Edition. John Wiley & Sons, New York, 1995.

- [32] C Villamizar. <http://brookfield.ans.net/omp/random-test-cases.html>.
- [33] JRL Webb. *Functions of Several Real Variables*. Ellis Horwood Limited, Chichester, 1991.
- [34] RW Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice-Hall, Englewood Cliffs, 1989.

Index

- algorithm
 - iterative, 83
- arrival rate, 5, 25, 38
- ATM network, 3
- bandwidth, 3
 - “infinite”, 3, 22
- bandwidth management, 1
- basic solution, 73
 - optimal, 73
- basis
 - optimal, 84
- Bertsekas-Gallager, 45
- Bertsekas-Gallager algorithm, 16–20, 41–43, 45, 46, 48–51
- bi-directionality, 22
- blocking probability, 1
- capacity
 - link, 5, 38
- capacity reservation, 3
- Cauchy-type steepest descent, 16
- Cisco, 1
- complexity, 15, 20
- concave function, 63–66, 77
- connection-oriented, 1
- connectionless, 1
- constraint, 70, 71, 71*f*, 72–74, 76, 77, 82
 - convex, 75
 - equality, 71, 72
 - inequality, 70–73, 81
 - linear, 70, 71*f*, 75, 76
 - positivity, 70, 73, 83
- convergence criterion, 86
- convex function, 13, 27, 42, 63–66, 75–77, 85
 - quadratic, 85
- convex set, 63, 65, 75, 76, 76*f*
- cost
 - incremental, 14, 15, 18
- cost function, 5, 7, 11, 21, 37
- cost rate vector, 8, 13–15, 18, 19, 27–33, 42–44
 - second derivative, 17, 19, 42
- CRLSP, 2
- decision variable, 12, 26, 70, 73, 84
- delay
 - link, 3
 - nodal, 3
- destination address, 2
- Dijkstra’s algorithm, 13, 15, 18, 30–33, 43, 44
- dual problem, 73
- EDR, 2
- egress node, 23
- egress packet, 23
- equilibrium, 41
- expected delay, 28
- expected waiting time, 40
- exponential distribution, 21, 37
- external traffic, 6, 38
- feasible region, 74–77
- FEC, 2, 7, 23, 24
- Fibonacci numbers, 87
- Fibonacci search, 14, 28, 87–89
- first derivative length, 9, 10
- flow, 7, 9, 12–14, 19, 23, 27, 28, 43, 46
 - extremal, 9
 - least-cost, 9

- total, 30, 31, 33, 44
- flow deviation, 3, 5–21, 26, 40, 41, 45, 46, 63, 67, 73*f*, 79
 - convergence, 45
- flow preservation, 7, 11
- flow vector, 10, 27–32, 43
 - feasible, 26–30, 35
 - initial, 30–33
 - least-cost, 28, 30
 - optimal, 28
- forwarding granularity, 2
- FR, 2
- Frank-Wolfe method, 13, 15, 26*f*, 67–86
- global minimum, 27
- golden section search, 14, 28, 33
- gradient projection method, 16, 18, 94–95
- gradient vector, 8, 11, 27, 69, 77, 78, 80, 83–85, 92
- Hessian matrix, 95
- hyperplane
 - normal, 77
 - supporting, 76, 77
- IBM, 1
- IETF, 2
- increment, 93
- incremental delay, 27–30
- independence assumption, 42
- infinitesimal flow, 9
- ingress node, 22, 23
- ingress packet, 23
- Internet, 1
- IP, 1
 - network, 1, 3
 - packet, 1, 3
- Ipsilon, 1
- iterative method, 91
- iterative step, 42, 91, 93–95
- Jacobian matrix, 92
- Jensen's inequality, 64, 65
- Kleinrock algorithm, 16, 40–43, 45–50
- label, 1, 2
 - global, 1
 - local, 1
- label assignment, 1
- label binding, 1
- label swapping, 1, 2, 23
- label switching, 1–3, 7
- label switching network, 5, 7
- Lagrangian function, 74, 76
- least-cost route, 14, 17
- LIB, 2
- line
 - supporting, 76
- line search, 14, 28, 29, 32, 33, 42, 84–87
- link capacity
 - “infinite”, 37
- link cost function, 26
- link delay, 3, 22, 37–40
- link delay function, 40
- link flow, 7, 8, 12, 26
- link flow vector, 7, 13–15, 18, 19, 40, 42, 43
 - feasible, 13, 14, 14*f*, 18, 41, 43
 - infeasible, 42
 - initial, 43–44
 - least-cost, 13–15, 18
 - optimal, 13, 14
- LSP, 2, 3, 7, 21–28, 30, 31, 35, 37, 38, 40, 46–50
 - cost, 28
 - least-cost, 28, 30–33, 42–44, 48, 50
 - optimal, 26, 40
- LSP arrival rate, 26
- LSP configuration, 45
- LSP discovery, 40
- LSP set, 46
- LSR, 1–3, 20, 21, 23, 25, 37
 - egress, 2
 - ingress, 1
- LSR load
 - maximum, 31

- LSR routing table, 2
- LU decomposition, 92
- matrix
 - symmetric, 68, 70, 79
- minimisation, 93
 - unconstrained, 93
- minimisation problem, 70, 93, 94
- minimum
 - global, 13
 - local, 65
- minimum slack, 42
- $M/M/1$, 25
- $M/M/1$ queue, 26
- $M/M/1$ queue, 40
- MPLS, 2, 21, 37
- MPLS network, 3, 7, 14*f*, 20, 26, 37
- network
 - symmetric, 34
- network cost, 8, 14
 - incremental, 13, 14
- network throughput, 1
- Newton's method, 16
- Newton-Raphson method, 91–93
- nodal arrival rates, 26
- nodal cost function, 26
- nodal delay, 3, 22, 24, 25, 37
 - expected, 25
- normalised length, 50
- objective function, 70, 71, 71*f*, 85
- optical fibre, 3
- optimal solution, 71
- optimisation, 70–78, 91
 - constrained, 94, 95
- optimisation problem, 5, 7, 26, 70, 71, 71*f*, 73, 74, 83, 87
 - convex, 74
 - linear, 71, 71*f*, 73, 83
 - non-linear, 7, 8, 26, 40
 - quadratic, 80
 - telecommunications, 73*f*
- optimum, 81, 83
 - global, 65
 - local, 65
- optmisation
 - unconstrained, 94
- packet arrival rate, 5–7, 23, 24, 27, 38, 39
- packet delay, 21, 24, 25, 37, 39
 - expected, 2, 3, 26, 37, 40, 42, 49
- packet flow, 3, 25, 37
- packet flow vector, 26
- packet length, 21, 37
- performance criterion, 2, 3
- Poisson process, 22, 38
- positive definite, 70
- positive ortant, 95
- positive semi-definite, 70, 79
- primal problem, 74
- priority queueing flag, 1
- profit rate, 3
- propagation delay, 22
- QoS, 1, 3, 20
- QoS management, 1
- QoS optimisation, 2
- quadratic form, 68–70
- queueing discipline, 25
- queueing mechanism, 21
- real inequality, 72
- revenue rate, 1, 3
- root finding problem, 93
- route, 6–10, 13, 14, 38
 - extremal, 9, 13, 16, 17
 - least-cost, 9, 10, 12, 13, 15, 16, 18, 19
- route cost, 8, 9
- route flow, 12, 13, 16, 17, 26
 - optimal, 10
- route flow vector, 7, 10, 11, 14*f*
 - feasible, 10
 - optimal, 9, 10
- routing
 - destination-based, 2

- explicit, 2
 - optimal, 1
- routing decision, 2
- routing pattern, 34
- saddle point, 74
- SDR, 2
- service class, 2, 21, 23, 25, 26
- service integrated, 23
- service integration, 25
- service rate, 25
- service separation, 25
- simplex algorithm, 71, 73, 83–85
- slack variable, 71, 81, 82
- solution
 - optimal, 9, 74, 80, 84, 85
- statistical equilibrium, 26
- steepest descent, 93
- steepest descent method, 93, 94
 - Cauchy-type, 93, 95
- step-size, 18–20, 43, 46
- subnetwork, 1, 2, 23
 - virtual, 1
- subnetwork device, 23
- switching, 7
- symmetric matrix, 69, 81, 82
- Taylor series, 91
- TDR, 2
- termination criterion, 14, 18
- Toshiba, 1
- transit node, 22, 23
- transit packet, 23
- unimodal function, 87
- vector space, 83, 84
- VNET, 3
- VP, 1
- VPCN, 1, 3
- waiting line, 25
- waiting time
 - expected, 25